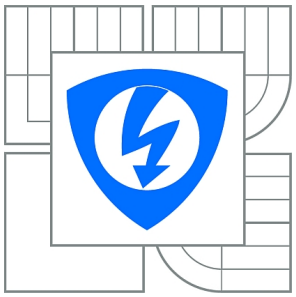
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁVRH A TESTOVÁNÍ STOCHASTICKÉ NAVIGACE V TRASI

DESIGN AND TESTING OF STOCHASTIC NAVIGATION IN TRAFFIC SIMULATOR TRASI

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH ERBEN

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, Ph.D.

SUPERVISOR

BRNO 2013

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Vojtěch Erben
Ročník: 2

ID: 115166
Akademický rok: 2012/13

NÁZEV TÉMATU:

Návrh a testování stochastické navigace v TRASI

POKYNY PRO VYPRACOVÁNÍ:

- seznamte se s dopravním simulátorem TRASI a základy teorie dopravních toků
- zpracujte stručný přehled algoritmů pro hledání průchodu orientovanými grafy
- implementujte vybraný algoritmus do TRASI
- navrhněte a popište vlastní algoritmus stochastické navigace
- implementujte algoritmus do TRASI včetně způsobu komunikace mezi vozy
- vyhodnoťte změny v dopravních tocích při použití stochastické navigace
- upravte stávající uživatelské rozhraní simulátoru

DOPORUČENÁ LITERATURA:

Šeda, M.: Teorie grafů. Skripta VUT FSI v Brně, 2003.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 20.5.2013

Vedoucí práce: Ing. Petr Honzík, Ph.D.
Konzultanti diplomové práce: Ing. Ondřej Hynčica

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabývá návrhem a implementací algoritmů hledání trasy v dopravním simulátoru TRASI. Tyto algoritmy jsou schopny naplánovat trasu vozidla sestávající ze seznamu křižovatek, kterými musí vozidlo projet. Dále se práce zabývá návrhem a implementací stochastické navigace včetně implementace komunikace mezi vozidly. Stochastická navigace na základě dopravní události navrhne několik alternativních tras jízdy. Z těchto tras, na základě informací o propustnosti dílčích cest navržených tras, náhodně (stochasticky) jednu vybere.

V úvodu práce je popsán dopravní simulátor TRASI, jeho uživatelské rozhraní a základní ovládání. Dále je popsána teorie dopravního toku na makroskopické i mikroskopické úrovni. Následuje popis algoritmů pro hledání průchodu v orientovaných grafech a jejich implementace v simulátoru. V další části práce je popsána komunikační vrstva zajišťující komunikaci vozidel a její implementace. Dále je popsán návrh a implementace stochastické navigace. V závěrečné kapitole je provedeno ověření funkčnosti simulátoru a testy jednotlivých algoritmů hledání trasy.

KLÍČOVÁ SLOVA

TRASI, simulátor, doprava, navigace, stochastická navigace, komunikace vozidel

ABSTRACT

The thesis deals with the design and implementation of routing algorithms in trafic simulator TRASI. These algorithms are capable of planning vehicle's route by giving a set of crossroads that vehicle needs to go through. Furthermore, this work deals with design and implementation of stochastic navigation including implementation of communication between vehicles. Stochastic navigation suggests several alternative routes based on a traffic event. From these routes is randomly (stochastically) choosen one based on information about the throughput of particular found routes.

In the introduction of this work is described the traffic simulator TRASI, it's user interface and basic control interface. Further is described theory of traffic flow on macroscopic and microscopic level, followed by the descripion of algorithms for oriented graphs traversal and their implementation in the simulator. In the following parts of this thesis is described communication layer, that takes care of the communication between vehicles, and it's implementation. Further is described design and implementation of stochastic navigation. In the final chapter is done verification of the functionality of the simulator and tests of particular routing algorithms.

KEYWORDS

TRASI, simulator, traffic, navigation, stochastic navigation, vehicles communication

ERBEN, Vojtěch *Návrh a testování stochastické navigace v TRASI*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2013. 67 s. Vedoucí práce byl Ing. Petr Honzík, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Návrh a testování stochastické navigace v TRASI“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Petru Honzíkovi, Ph.D. za rady, trpělivost, ochotu vždy pomoct a přátelský přístup. Také bych chtěl poděkovat svým rodičům za morální a finanční pomoc.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ALGORITMŮ

ÚVOD

Cílem diplomové práce je navrhnout a implementovat nový způsob navigace vozidel. V rámci řešení práce je potřeba se seznámit s již existujícím dopravním simulátorem Traffic simulator (TRASI), který slouží k modelování silničního provozu. Pro účely navigace bude zpracován stručný přehled algoritmů pro hledání průchodu orientovanými grafy. Vhodný algoritmus bude implementován v simulátoru TRASI. Navigace bude schopna naplánovat trasu vozidla z výchozího bodu do cíle. Plán cesty bude sestávat ze seznamu křižovatek, kterými je třeba projet až do koncového bodu (cíle) cesty. Dále budou implementovány komunikační vrstvy typu Car-to-Car a Car-to-Infrastructure sloužící ke komunikaci vozidel, které si budou předávat zprávy o dopravních událostech (zácpa, zpomalení plynulosti provozu) a na jejich základě bude upravována trasa ostatních vozidel.

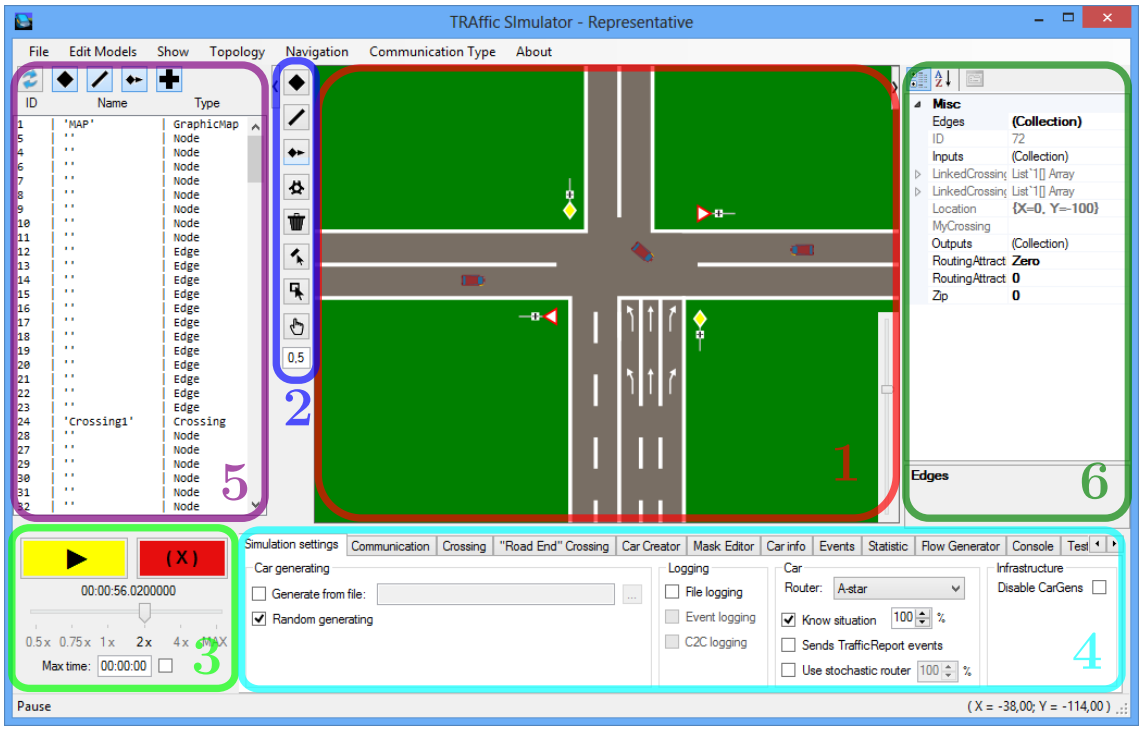
Dalším úkolem je navrhnout a implementovat nový typ navigace, tzv. stochastickou navigaci. Ta na základě dopravní události navrhne několik alternativních tras jízdy. Z těchto tras, na základě informací o propustnosti dílčích cest navržených tras, náhodně (stochasticky) vybere jednu z alternativních tras. Tento náhodný výběr by měl zabránit vzniku kolon způsobených výběrem stejné trasy u všech ostatních vozidel projíždějících danou oblastí.

Funkčnost simulátoru TRASI bude verifikována na základě teoretických předpokladů plynoucích z teorie dopravních toků. Bude také porovnán běžný algoritmus navigace se stochastickou navigací.

1 DOPRAVNÍ SIMULÁTOR TRASI

Simulátor TRASI [7] je simulátor dopravního provozu vyvíjený v rámci projektu Grantové agentury České republiky (GA ČR) [3]. Simulátor obsahuje grafické uživatelské rozhraní (GUI), umělou inteligenci aut včetně pravidel silničního provozu, algoritmy hledání trasy a komunikační vrstvy pro předávání zpráv mezi auty. Umělá inteligence vyhodnocuje vzdálenost každého auta od ostatních, řeší přejíždění z pruhu do pruhu a realizuje průjezd křižovatkami. Každé auto se na základě umělé inteligence rozhoduje zda bude zrychlovat nebo zpomalovat a nastavuje natočení kol.

Během řešení diplomové práce byly implementovány algoritmy hledání trasy, systém hlášení dopravních událostí a jejich předávání mezi vozidly. Také byl implementován nový způsob hledání trasy zvaný stochastická navigace, viz kapitola 7. Uživatelské rozhraní bylo vylepšeno (viz obrázek 1.1) a byl implementován nový způsob ovládání simulátoru a pohybu ve virtuálním prostředí, viz kapitola 2.



Obr. 1.1: Grafické uživatelské rozhraní (GUI)

GUI se skládá z těchto součástí:

- Grafický výstup simulace (1)
- Nástrojová lišta pro tvorbu a editaci mapy (2)
- Ovládání běhu simulace (3)
- Záložky s nastavením (4)

- Seznam objektů v simulaci (5)
- Detail vybraného objektu (6)
- Aplikační menu (panel v horní části okna)
- Stavový panel (panel v dolní části okna)

1.1 Simulační prostředí

Simulační prostředí je vytvářeno pomocí těchto základních prvků:

- Uzel (`Node`)
- Generátor aut (`CarGen`)
- Hrana (`Edge`)
- Křižovatka (`Crossing`)
- Ukončení silnice (`RoadEnd`)

Základní prvkem topologie je uzel reprezentovaný třídou `Node`. Uzel je místo, kde se setkávají dvě nebo více hran. Obsahuje seznam hran, které jsou s ním propojené.

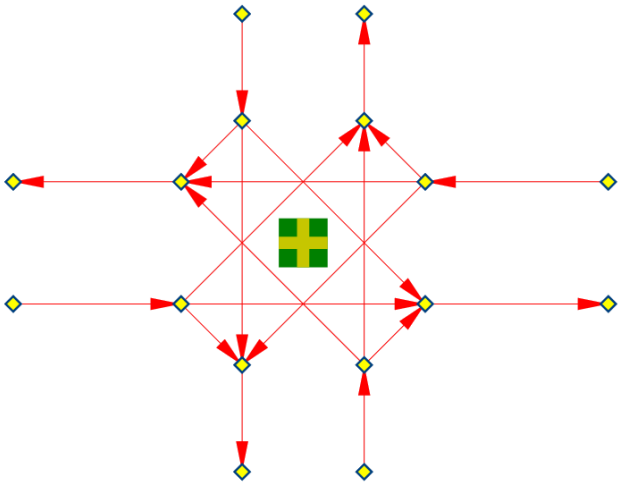
Speciálním typem uzlu je generátor aut reprezentovaný třídou `CarGen`. Tento uzel slouží k vytváření aut v simulaci. Auta mohou být vytvářena periodicky se zvoleným časovým rozestupem nebo individuálně. V případě periodického vytváření generuje uzel auta se stejným typem navigace s výchozími parametry. Individuální seznam generovaných aut se načítá ze speciálního souboru a umožňuje definovat čas vytvoření, parametry a typ navigace každého vytvořeného auta zvlášť.

Spojnice mezi uzly se nazývají hrany a jsou reprezentovány třídou `Edge`. Každá hrana má určenou orientaci (auto smí projíždět pouze v jednom směru) a může se skládat z libovolného počtu bodů, ale počáteční a koncový bod jsou vždy uzly.

Křižovatka je objekt složený z hran a uzlů a je reprezentován třídou `Crossing`. Na obrázku 1.2 je grafická reprezentace křižovatky v GUI se čtyřmi vstupy a čtyřmi výstupy. Každý vstup je spojen se všemi výstupy křižovatky (červené čáry reprezentující hrany `Edge`). Žluto-modré čtverce znázorňují uzly (`Node`). Kromě této křižovatky je možné volit mezi nejrůznějšími kombinacemi počtu vstupů a výstupů křižovatky a je možné vytvářet i kruhový objezd.

Speciálním typem křižovatky je ukončení silnice. Tato křižovatka má několik vstupů a výstupů pouze v jednom směru. Výstupy jsou tvořeny uzly typu `CarGen`, vstupní uzly jsou běžného typu a přijíždějící auta se v nich ztrácí.

V uživatelském rozhraní simulátoru TRASI je možné přidávat nebo mazat všechny výše popsané druhy uzlů, hran a křižovatek. Při jejich vytváření jim lze nastavit některé jejich parametry. Tatko vytvořené mapy lze ukládat do souboru jako tzv. workspace.
Z těchto základních prvků je možné vytvořit:



Obr. 1.2: Grafická reprezentace křižovatky v simulátoru TRASI s vyznačenou orientací hran

- kruhový objezd,
- mimoúrovňové křížení (vozidla jsou při průjezdu pod mostem v grafickém výstupu skryta),
- jednosměrné silnice,
- nájezdy/sjezdy,
- světelné křižovatky s nastavitelným časováním,
- křižovatky s určením přednosti (hlavní silnice)

Základní prvky nepodporují:

- křižovatky s více výstupními pruhy (pouze více vstupních pruhů),
- křižovatky propojující více než 4 silnice,
- vytváření jiných dopravních prostředků (např. tramvaje)
- simulaci přechodů pro chodce
- speciální vozidla (záchranná služba, hasičské vozidla, policejní vozidla aj.)

1.2 Umělá inteligence automobilů

Umělá inteligence zajišťuje autonomní chování automobilů v simulaci. Jejím úkolem je dodržování pravidel silničního provozu a realizace navigace aut mezi jednotlivými křižovatkami navržené trasy.

Každé auto má definován rozvor náprav, maximální hodnotu akcelerace a decelerace a maximální rychlost. Umělá inteligence každého auta v každém simulačním kroku vyhodnocuje zda je potřeba změnit natočení kol nebo zda je třeba zrychlit nebo zpomalit. Auta nemusí vždy přesně dodržovat maximální povolenou rychlost

na dané silnici, ale mohou ji i překročit nebo jet pomaleji. Dodržování rychlosti je možné měnit v nastavení (parametr *speed coeficient*). Při jízdě auto určuje natočení volantu podle pozice uzlu (`Node`) nebo bodu na hraně, do kterého směřuje. Díky tomu auto projíždí plynule i přes ostré hrany.

Automobil může u víceproudých silnic přejíždět mezi pruhy tak, aby dorazil na správný vstup křižovatky (např. odbočovací pruh). Pokud se více hran sbíhá do jedné hrany, zařazují se auta do společného pruhu podle strategie zip (střídavě vjíždí do společného pruhu auto z prvního a druhého pruhu). Na křižovatce se auto řídí buď světelnou signalizací, hlavní silnicí nebo předností zprava. Při vyhodnocování možnosti průjezdu křižovatkou je brána v potaz rychlost a vzdálenost auta blížícího se z druhé silnice a míra „odvahy“ rozhodujícího se auta. Faktor odvahy je možné měnit v nastavení (parametr *precaution*).

Při průjezdu automobilu křižovatkou se jeho směr určí na základě trasy, která je tvořena seznamem křižovatek. Umělá inteligence určí správnou hranu, pro které má auto jet tak, aby se dostalo do následující křižovatky, kterou má podle svojí trasy projet. Pokud žádná z výstupních cest z křižovatky nevede do křižovatky, do které má auto jet, auto zvolí náhodný směr a požádá navigaci o novou trasu. Trasa se pro každé auto vytvoří při jeho vzniku v simulaci tak, že je buď náhodně vybrána jedna křižovatka typu `RoadEnd` jako cílová nebo je cílová křižovatka zadána v konfiguračním souboru. Router (plánovač trasy) poté najde cestu do dané křižovatky. Výstupem routeru je seznam křižovatek, které musí auto projet.
Automobily mohou:

- přejíždět mezi pruhy ve stejném směru (pouze v důsledku zařazení do správného odbočovacího pruhu),
- řadit se metodou zip při zúžení silnice do jednoho pruhu,
- měnit svou trasu na základě obdržení zprávy o dopravní situaci,
- nedodržovat maximální povolenou rychlost na základě parametru *speed coeficient*,
- mít různé parametry jako rozvor náprav, maximální rychlost, akceleraci, deceleraci, faktor odvahy

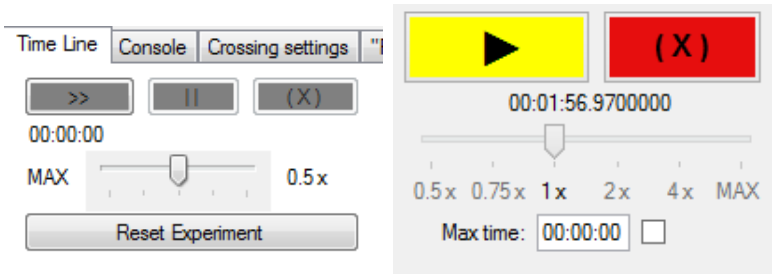
Automobily nemohou:

- předjíždět jiné automobily přejetím do jiného pruhu,
- zastavit na předem určeném místě (parkování),

Podrobnější popis fungování simulátoru TRASI je možné nalézt v [7].

2 UŽIVATELSKÉ ROZHRANÍ

Během řešení práce bylo upraveno stávající uživatelské rozhraní (viz obrázek 2.2). Přepracováno bylo ovládání běhu simulace (levá spodní část okna), které nyní sestává ze dvou tlačítek (spuštění/pozastavení simulace a reset simulace), posuvníku ovládání rychlosti simulace a pole umožňující nastavit časový limit běhu simulace. Ovládání simulace oproti původní verzi není součástí lišty záložek, a je tedy neustále přístupné, viz obrázek 2.1.



(a) Původní ovládání (b) Nové ovládání

Obr. 2.1: Ovládání běhu simulace

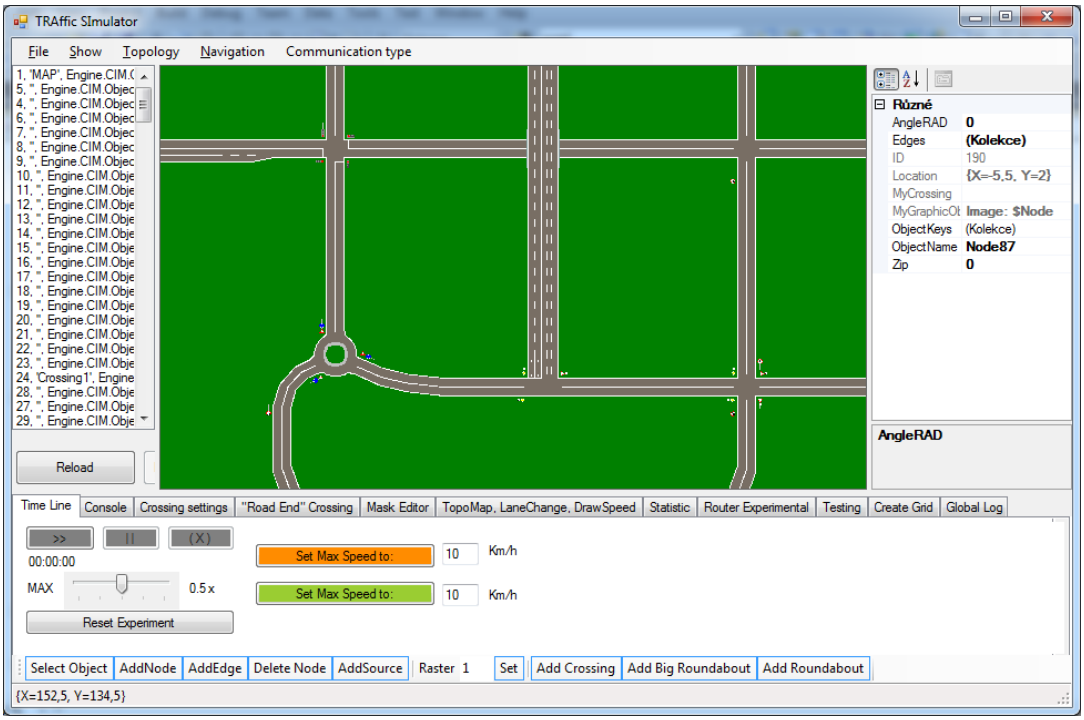
Dále byla upravena nástrojová lišta pro tvorbu mapy. Z původní pozice ve spodní části okna byla přesunuta na levou stranu okna. Byly přidány nové nástroje a vytvořeny ikony nástrojů včetně ukazatelů myši při jejich aktivaci. Popis nástrojů je možné nalézt v kapitole 2.2.

Některé původní záložky byly sloučeny a byly přidány následující nové záložky:

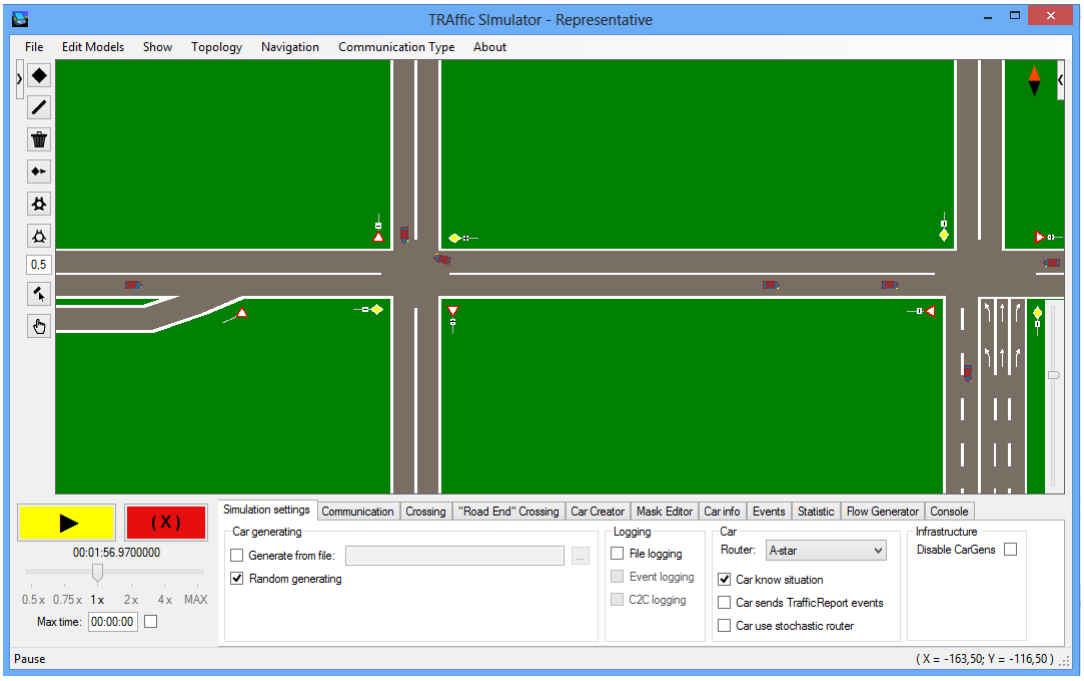
- Simulation settings
- Communication
- Car creator
- Car info
- Events
- Statistic
- Flow generator

Popis funkce těchto záložek přesahuje rozsah této práce a z toho důvodu nebude dále rozebírán.

Také byl vytvořen nový dialog nastavení (viz obrázek 2.3), do kterého byly přesunuty některé volby z původní lišty záložek. Dialog nastavení umožňuje nastavit výchozí parametry vozidel, komunikační vrstvy, záznamů, vzhledu simulátoru, rychlosti vykreslování, délku simulačního kroku a volby importu z OpenStreetMap [8].

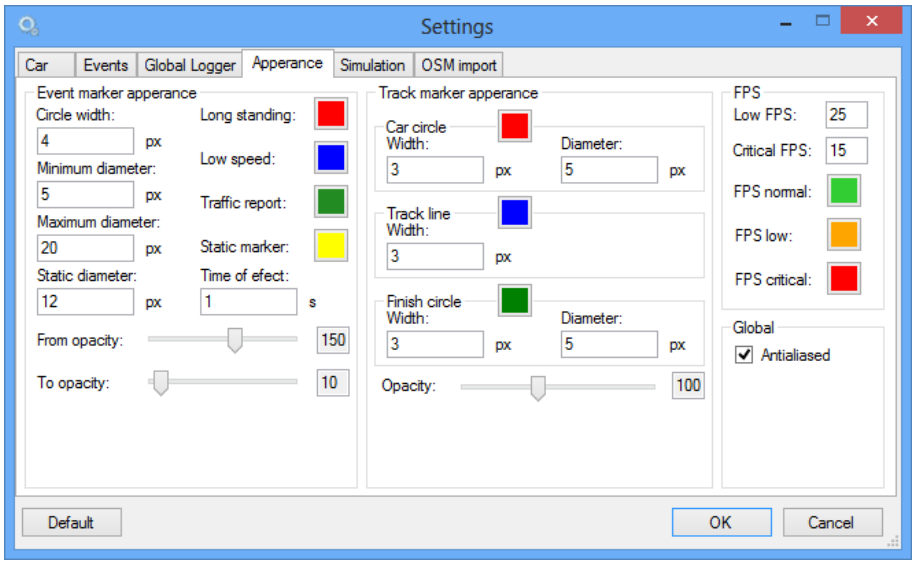


(a) Původní grafické uživatelské rozhraní



(b) Nové grafické uživatelské rozhraní

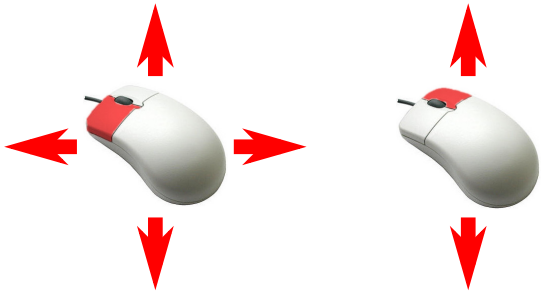
Obr. 2.2: Grafické uživatelské rozhraní simulátoru TRASI



Obr. 2.3: Dialog nastavení simulátoru

2.1 Základní ovládání

Následující popis tlačítek se týká operací v grafickém výstupu simulace. Základní pohyb v simulaci pomocí myši je na na obrázku 2.4.



(a) Posun (b) Přiblížení

Obr. 2.4: Základní ovládání simulátoru

Levé tlačítko myši

- V simulaci je možné se pohybovat stiskem levého tlačítka myši a jejím posunem.
- Stiskem je možné vyvolat akci nástroje vybraného v nástrojové liště.
- Pokud je zobrazen detail vybraného objektu, kliknutím je skryt.

Pravé tlačítko myši

- Kliknutím je možné vybrat objekt v simulaci (kromě vozidla). Po kliknutí se automaticky zobrazí panel s detailem vybraného objektu. Přidržením klávesy *Ctrl* je možné vybrat více objektů najednou.
- Přidržením a pohybem myši nahoru a dolů je možné rychle měnit úroveň přiblížení. V tomto režimu je na obrazovce zobrazen zaměřovač, který určuje bod kam se přibližuje/oddaluje. Pokud je navíc stisknuta klávesa *Shift* je možné rychle měnit úhel natočení.

Střední tlačítko myši

- Stisknutím je možné vybrat objekt auta v simulaci. Detailní informace o autě je pak možné sledovat v panelu s detailem vybraného objektu. Tato akce také spolupracuje se záložkami *Car info* a *Events* ve spodní části okna.
- Pokud je stisknuta klávesa *Shift* je možné stiskem nastavit výchozí natočení pohledu.

Kolečko myši

- Rotací je možné měnit úroveň přiblížení.
- Rotací při stisknuté klávese *Shift* je možné měnit natočení pohledu.

Mezerník

- Při tvorbě hran ukončuje danou hranu, viz kapitola 2.2.
- Pokud není vybrán žádný nástroj z nástrojové lišty spouští/pozastavuje běh simulace.

Delete

- Smaže vybrané objekty ze simulace. Objekty je možné vybírat pravým tlačítkem myši, nástrojem výběru hran nebo v seznamu objektů v simulaci (obrázek 1.1, část 5).

Ctrl

- Držením klávesy je možné vybírat více objektů najednou pravým tlačítkem myši nebo nástroji pro výběr oblasti a hran.

2.2 Tvorba mapy

Pro vytvoření nové mapy je potřeba vytvořit nový workspace z nabídky *File* → *New workspace*. V takto vytvořeném workspace je možné přidávat základní objekty

topologie pomocí nástrojové lišty (viz obrázek 1.1, část 2).

Pro všechny nástroje platí: stiskem levého tlačítka myši se vyvolá akce spojená s daným nástrojem (například se objekt přidá na zvolené místo v simulaci). Držením levého tlačítka myši a posunem je možné měnit pozici v simulaci bez vyvolání akce nástroje. Nástroje je možné zrušit stiskem klávesy *ESC*. Popis ovládání daného nástroje se zobrazuje ve stavové liště programu. Význam jednotlivých tlačítek a práce s nimi je následující:

- Slouží pro přidání uzlu (`node`). Stiskem klávesy *mezerník* se automaticky zvolí nástroj kreslení hran.
- Slouží pro vytváření hran (`edge`) mezi uzly. Prvním stiskem levého tlačítka myši se vybere nejbližší uzel a zvolí se jako začátek nové hrany. Při pohybu myši je zobrazována aktuální podoba hrany. Dalším kliknutím levého tlačítka myši se hrana ukotvuje, čímž je možné vytvářet zalomení hrany. Pro ukotvení hrany k cílovému uzlu slouží klávesa *mezerník*, po jejímž stisku je nalezen nejbližší uzel vůči aktuální pozici myši a je zvolen jako koncový uzel hrany. Opakováním tohoto postupu je možné přidávat další hrany.
- Slouží k přidávání generátorů aut. Jedná se o speciální typ uzlu, který generuje automobily. Stiskem klávesy *mezerník* se automaticky zvolí nástroj kreslení hran. Pro tvorbu zakončení silnice je vhodnější záložka *„Road End“ Crossing* (viz dále).
- Slouží k přidávání kruhových objezdů. Stiskem klávesy *mezerník* se automaticky zvolí nástroj kreslení hran.
- Slouží k mazání objektů. Tímto nástrojem je možné mazat všechny objekty kromě hran. Tento nástroj je možné vyvolat stiskem klávesy *Delete* pokud v není vybrán žádný jiný nástroj.
- Slouží k výběru hrany (`edge`) v simulaci. Vybírá se nejbližší hrana, která se zbarví modrou barvou. Je možné vybrat více hran najednou přidržením klávesy *Ctrl*. Stiskem pravého tlačítka myši je možné vybírat ostatní objekty v simulaci (lze kombinovat s přidržením *Ctrl* klávesy). Po výběru hrany je možné stiskem klávesy *Delete* smazat vybrané objekty.
- Slouží k hromadnému výběru všech objektů v označené oblasti. Z oblasti jsou vybrány pouze objekty povolené v levém panelu v horní části pro filtrování zobrazených objektů. Levým tlačítkem myši je zvolen nejdříve první bod výběru, druhým stiskem druhý bod. Je možné kombinovat více výběrů přidržením klávesy *Ctrl* během označení druhého bodu dalšího výběru.
- Slouží k posunu křižovatek (`crossings`) a uzlů (`node`). Stiskem levého tlačítka myši se vybere nejbližší uzel nebo křižovatka a daný objekt je přichycen k ukazateli myši. Pohybem myši je možné objekt posouvat. Přidržením klávesy *Ctrl* a posunem kolečka myši je možné daným objektem rotovat. Přidržením klávesy

Ctrl a stiskem středního tlačítka myši se vrátí rotace na původní hodnotu. Stiskem klávesy *ESC* se objekt vrátí na původní pozici a původní natočení.

Textové pole v nástrojové liště slouží k nastavení velikosti gridu, ke kterému je přichytáván kurzor myši. Ke tvorbě mapy slouží také záložky (viz obrázek 1.1, část 4):

Záložka *„Road End“ Crossing* slouží k tvorbě zakončení silnic. Jedná se o speciální typ křižovatky, která obsahuje generátory aut. Při přidávání této křižovatky je možné měnit její rotaci přidržením klávesy *Ctrl* a rotací kolečka.

Záložka *Crossing* slouží k přidávání křižovatek do simulace. Je možné nastavit počet silnic a odbočovacích pruhů. Je možné měnit její rotaci přidržením klávesy *Ctrl* a rotací kolečka.

3 TEORIE DOPRAVNÍHO TOKU

Aby bylo možné navrhnout silnice a auta, které umožňují a usnadňují osobní dopravu, je potřeba modelovat chování jednotlivých aut a jejich řidičů v jednom pruhu a také skupiny aut v jednom nebo více pruzích. Toto modelování je zaměřeno především na interakce mezi auty.

Dopravní tok je možné zkoumat ze dvou úrovní. Makroskopické modelování [2] provozu předpokládá dostatečně velký počet aut v daném pruhu nebo na silnici, kde může být každý tok aut chápán jako proudění kapaliny v potrubí. Makroskopické modely jsou vyjádřeny třemi proměnnými:

Dopravní tok (rate of flow) počet aut, které projedou daným bodem za jednotku času

Rychlost dopravního toku (speed of the traffic flow) ujetá vzdálenost za jednotku času.

Dopravní hustota (traffic density) počet aut ve zvoleném úseku definované délky.

Grafická závislost vztahu mezi rychlostí a hustotou se nazývá fundamentální diagram.

Druhou úrovní pohledu je mikroskopické modelování [2], které se zabývá interakcí jednotlivých aut v jízdním pruhu. Mikroskopický model popisuje jak reagují auta v koloně na vedoucí auto kolony modelováním jeho akcelerace jako funkce různých podnětů. Může to být například vzdálenost mezi vedoucím a dalším autem v koloně, relativní rychlost dvou aut nebo reakční čas řidiče v druhém vozidle. Existuje spousta variant těchto modelů, které mohou být použity na sestavování křivek závislosti rychlosti dopravního toku a dopravní hustoty podpořené daty z reálného provozu. To dovoluje dopravním expertům modelovat a pochopit kapacitu silnic jako funkci rychlosti dopravního toku a dopravní hustoty. Mikroskopické modely jsou také používány jako podpora modelování řízení dopravy pro implementaci řídící strategie, která dovolí udržet vysoký dopravní tok při jeho současné vysoké rychlosti.

3.1 Makroskopický pohled

3.1.1 Vztah mezi rychlostí dopravy a dopravní hustotou

Vztah mezi rychlostí dopravy a dopravní hustotou je možné pozorovat v reálném provozu. Pokud je hustota malá, rychlost provozu roste a naopak. Můžeme tedy

předpokládat že mezi rychlostí dopravy a dopravní hustotou je přímý vztah:

$$v \ = v(\rho) \tag{3.1}$$

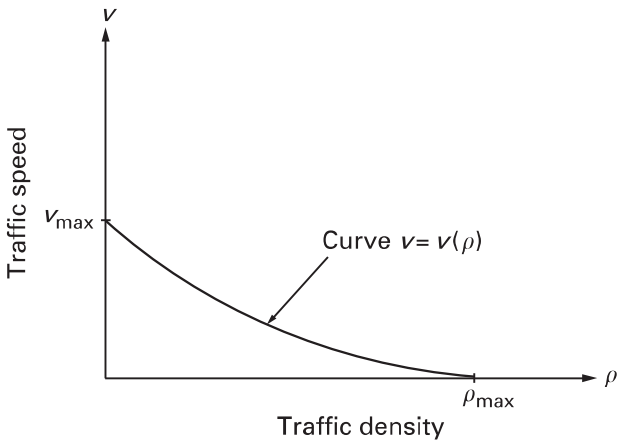
Na základě tohoto pozorování je možné očekávat že řidiči pojedou nejrychleji (v_{max}) když bude dopravní hustota nejnižší ($\rho \to 0$). Rychlost klesá s tím jak roste hustota provozu. Nakonec se provoz zastaví ($v = 0$) na maximu hustoty (ρ_{jam}). Tento jev je možné matematicky shrnout takto:

$$v(\rho = 0) \ = \ v_{max}, \tag{3.2a}$$

$$\frac{\mathrm{d}v}{\mathrm{d}\rho} \ \leq \ 0, \tag{3.2b}$$

$$v(\rho = \rho_{jam}) \ = \ 0. \tag{3.2c}$$

Tuto závislost je možné také zobrazit graficky pomocí obecné křivky zobrazené na obrázku 3.1. Přesný tvar křivky není v tento okamžik známý, pouze koncové body a směr sklonu [2].



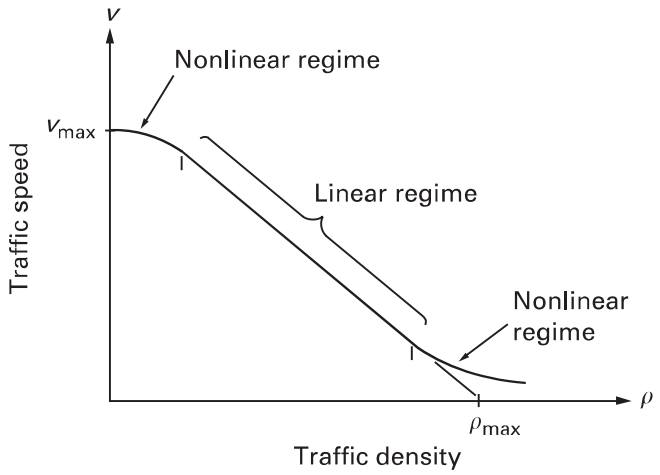
Obr. 3.1: Závislost dopravní rychlostí na hustotě (Zdroj: [2]).

Základní předpoklady modelu nezahrnují všechny možnosti, ačkoliv zkušenosti ukazují, že rovnice (3.1) a (3.2) odpovídajícím způsobem odrážejí chování provozu. Modely analyzující vztah rychlosti dopravy a dopravní hustoty odrážejí spíše lidské chování než zákony mechaniky, jelikož zachycují jak řidiči reagují na vnější podněty. Ve skutečnosti tento vztah rychlosti a hustoty (jako rovnice (3.1)) vychází z empirických dat.

Pro zpřesnění některých z těchto kvalitativních myšlenek je možné uvažovat následující vztah mezi rychlostí a hustotou dopravy [2]:

$$v(\rho) = v_{max}\left(1 - \frac{\rho}{\rho_{jam}}\right) \tag{3.3}$$

Tento vztah splňuje všechny podmínky rovnic (3.2). Ve studiích prováděných pro tunely Lincoln v Holandsku a Queens-Midtown v New Yorku se lineární vztah mezi rychlostí a hustotou dopravy ukázal jako velmi dobrá aproximace pro střední (dominantní) část křivky z nasbíraných empirických dat [2]. Tato křivka je zobrazena na obrázku 3.2.



Obr. 3.2: Jiná závislost dopravní rychlosti na hustotě, založená na výsledcích, které se často objevují u dopravních systémů. Jsou zde vyznačeny stejné koncové body a je dodržena podmínka klesající křivky, což ukazuje že je možné významné části křivky modelovat pomocí vztahu rychlosti a hustoty. (Zdroj: [2])

3.1.2 Vztah mezi dopravním tokem a dopravní hustotou

Z pohledu dopravního inženýra, který navrhuje dopravní infrastrukturu (silnice, nájezdy, dopravní značení a signalizace atp.), je nejdůležitější proměnnou kapacita (nebo maximální dopravní tok), kterou musí dopravní systém pojmout, což vyjadřuje její dopravní tok $q(x,t)$ [2]. Pro makroskopický model je možné předpokládat, že je rychlost homogenní, což znamená že nezáleží na pozici na silnici x ani na čase t. Poté je možné napsat $v = v(\rho)$. Vztah mezi dopravním tokem q a hustotou ρ je:

$$q(\rho) = \rho \cdot v(\rho) \tag{3.4}$$

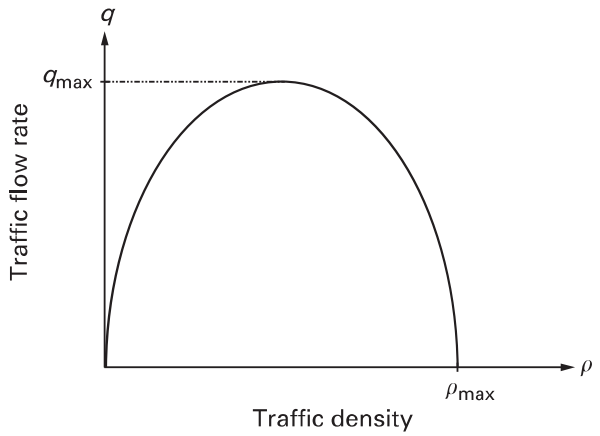
Dopravní tok tedy závisí pouze na hustotě ρ.

Nyní je možné rozšířit analýzu závislosti mezi dopravní rychlostí a hustotou dopravy o vztah mezi dopravním tokem a hustotou. Nejvyšší rychlosti v_{max} řidiči dosahují, když je hustota dopravy nejnižší ($\rho = 0$). Z rovnice (3.4) vyplývá $q(\rho = 0) = 0$, tedy že dopravní tok je nulový. Obdobně když se doprava zastaví na její maximální hustotě ($v(\rho_{jam}) = 0$), z rovnice (3.4) vyplývá, že je dopravní tok opět nulový: $q(\rho_{jam}) = \rho_{jam} \cdot v(\rho_{jam}) = 0$. Dopravní tok musí být kladný pro všechny hodnoty

hustoty ($0 < \rho < \rho_{jam}$) a musí dosáhnout svého maxima q_{max} někde v tomto intervalu. Sklon křivky je dán vztahem:

$$\frac{dq}{d\rho} = v(\rho) + \rho\frac{dv}{d\rho} \tag{3.5}$$

Na základě těchto kvalitativních poznatků je možné vytvořit obecnou křivku, která je na obrázku 3.3 a nazývá se fundamentální diagram dopravního toku. Stejně jako u obrázku 3.1 není znám přesný tvar této křivky.



Obr. 3.3: Závislost dopravního toku na hustotě (Zdroj: [2]).

Pokud provedeme substituci rovnice (3.3) do rovnice (3.4) dostaneme vztah pro dopravní tok jako funkci hustoty provozu, která je parabolická:

$$q(\rho) = v_{max}\left(\rho - \frac{\rho^2}{\rho_{jam}}\right) \tag{3.6}$$

Maximální dopravní tok nastane, když bude sklon této křivky nulový:

$$\frac{dq(\rho)}{d\rho} = v_{max}\left(1 - \frac{2\rho}{\rho_{jam}}\right) = 0 \tag{3.7}$$

Rovnice (3.7) ukazuje že maximum dopravního toku leží ve střední části fundamentálního diagramu, v bodě $\rho = \rho_{jam}/2$ a jeho hodnota je:

$$q_{max} = \frac{1}{4}\rho_{jam}v_{max} \tag{3.8}$$

3.2 Mikroskopický pohled

Mikroskopický model popisuje jak se vzájemně ovlivňují páry aut v koloně. Lineární model odpovídá reakcím řidiče na relativní rychlost auta před ním. Každé auto

v simulaci je identifikováno diskrétní souřadnicí, která se v čase mění. Poté je pozice n-tého vozidla označena jako $x_n(t)$. Reakce (akcelerace) řidiče druhého auta (n+1) je:

$$\frac{\mathrm{d}^2 x_{n+1}(t)}{\mathrm{d}t^2} = -K_p \left(\frac{\mathrm{d}x_{n+1}(t)}{\mathrm{d}t} - \frac{\mathrm{d}x_n(t)}{\mathrm{d}t} \right) \tag{3.9}$$

Parametr K_p je citlivost řidiče a je konstantní. Rovnici je možné integrovat:

$$\frac{\mathrm{d}x_{n+1}(t+T)}{\mathrm{d}t} = -K_p(\mathrm{d}x_{n+1}(t) - \mathrm{d}x_n(t)) + C_{n+1} \tag{3.10}$$

Kde T je zpoždění reakce řidiče a C_{n+1} je libovolná konstanta s rozměrem rychlosti. Za předpokladu, že mají všechna auta stejnou délku, je možné definovat hustotu aut na silnici ρ jako:

$$\rho = \frac{L_R}{N_R} = \frac{1}{x_n(t) - x_{x+1}(t)} \tag{3.11}$$

Kde L_R je délka úseku silnice a N_R je počet aut v daném úseku. Za předpokladu, že všechna auta pojedou stejnou rychlostí v, můžeme rychlost vyjádřit jako:

$$v \ = \frac{K_p}{k} + C \tag{3.12}$$

Nyní je možné ze vztahu mezi hustotou ρ a dopravním tokem q:

$$q = kv \tag{3.13}$$

a z naměřených dat [2] určit, že maximální rychlost v_{max} nastává, když je hustota blízko nule a klesá s tím jak hustota roste. Při maximální hustotě ρ_{jam} auta v koloně stojí a jejich rychlost je tedy nulová. Odtud je možné určit konstantu C. Ze vztahu mezi rychlostí a hustotou je možné psát:

$$v(p) = \begin{cases} v_{max} & \rho < \rho_{crit} \\ K_p \left(\dfrac{1}{\rho} - \dfrac{1}{\rho_{jam}} \right) & \rho \geq \rho_{crit} \end{cases} \tag{3.14}$$

kde koeficient K_p je parametr citlivosti a kritická hustota ρ_{crit} je:

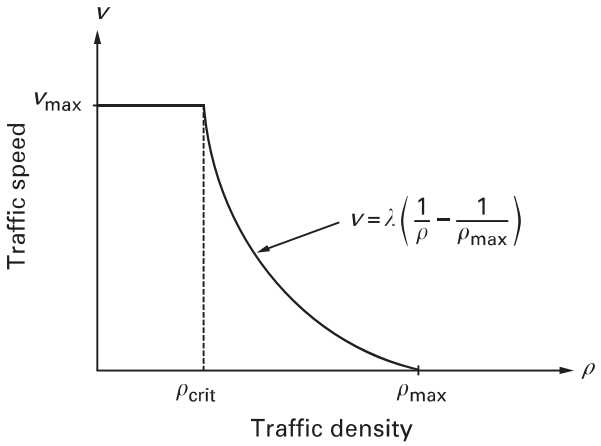
$$\rho_{crit} = \left(\frac{v_{max}}{K_p} + \frac{1}{\rho_{jam}} \right)^{-1} \tag{3.15}$$

Grafický průběh je na obrázku 3.4. Dopravní tok pak je:

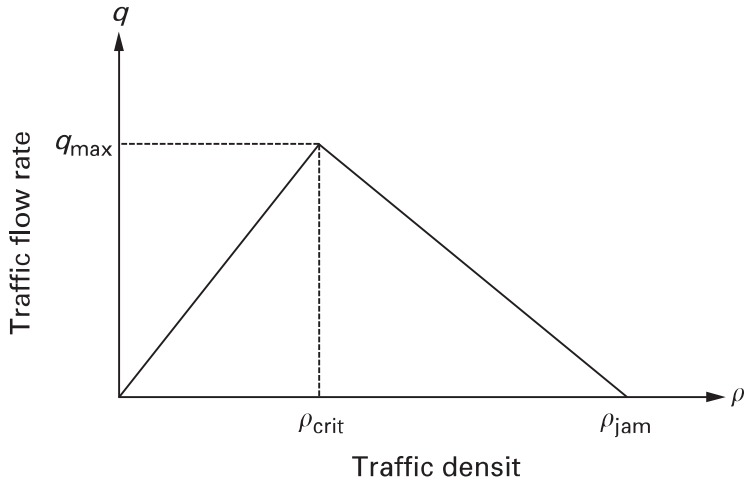
$$q(p) = \begin{cases} \rho v_{max} & \rho < \rho_{crit} \\ K_p \left(1 - \dfrac{\rho}{\rho_{jam}} \right) & \rho \geq \rho_{crit} \end{cases} \tag{3.16}$$

Dopravní tok (viz obrázek 3.5) lineárně narůstá s hustotou a dosahuje maximální hodnoty když $\rho = \rho_{crit}$:

$$q_{max} = q(\rho_{crit}) = \rho_{crit} v_{max} = K_p \left(1 - \frac{\rho_{crit}}{\rho_{jam}} \right) \tag{3.17}$$



Obr. 3.4: Závislost rychlosti dopravního toku na hustotě z mikroskopického pohledu. (Zdroj: [2])

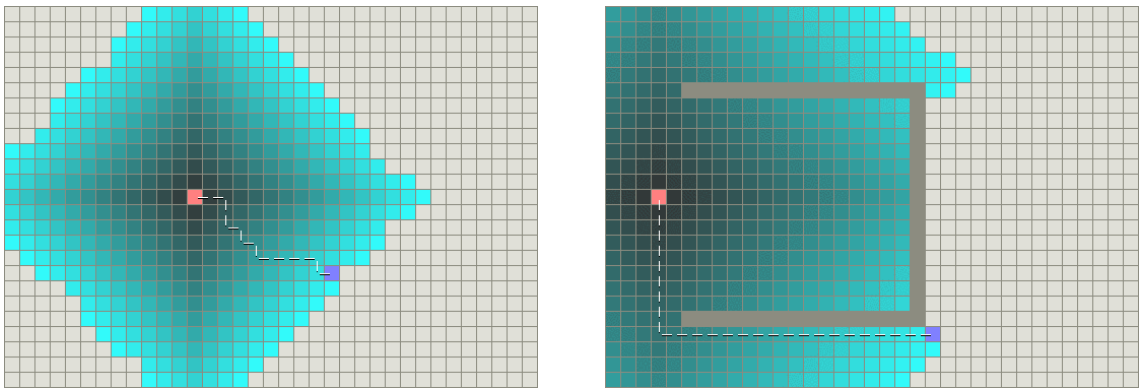


Obr. 3.5: Závislost dopravního toku na hustotě z mikroskopického pohledu. (Zdroj: [2])

4 ALGORITMY PRO HLEDÁNÍ PRŮCHODU V ORIENTOVANÝCH GRAFECH

4.1 Dijkstrův algoritmus

Dijkstrův algoritmus [9] pracuje tak, že prohledává vrcholy orientovaného grafu počínaje startovním vrcholem. Opakovaně zkoumá zatím neprověřené uzly ležící nejblíže ke startovnímu uzlu a ukládá si je do sady prověřených uzlů. Algoritmus prohledává všechny uzly od startovního uzlu, dokud nenalezne cíl.



(a) Průběh hledání cesty bez překážky

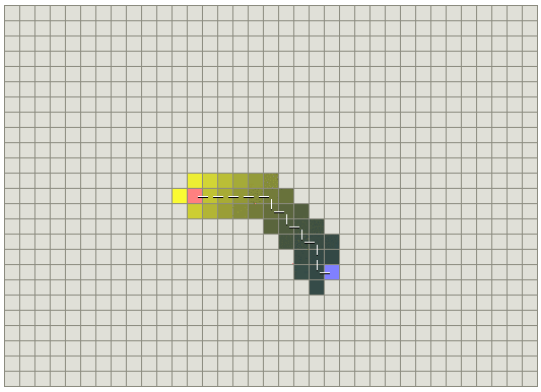
(b) Průběh hledání cesty s konkávní překážkou

Obr. 4.1: Průběh hledání cesty pomocí Dijkstrova algoritmu (zdroj: [9])

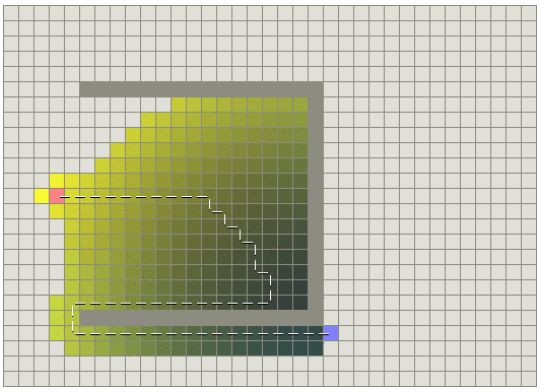
Na obrázku 4.1 je průběh prohledávání prostoru. Růžový bod je start a fialový je cíl. Světlejší oblasti jsou nejvzdálenější prohledané oblasti od startu a tvoří hranici prohledávání. Dijkstrův algoritmus je konečný, je tedy zaručeno, že nalezne nejkratší cestu do cíle. Obrázek 4.1b ukazuje průběh prohledávání v prostoru s konkávní překážkou. Dijkstrův algoritmus prohledá značnou část prostoru, ovšem je zaručeno že nalezne nejkratší cestu.

4.2 Algoritmus Best-first

Tento algoritmus [9] pracuje obdobně jako Dijkstrův algoritmus, přidává ovšem odhad vzdálenosti uzlu od cíle zvaný heuristika. Při prohledávání dává přednost uzlům blíže k cíli místo uzlům blíže ke startu. U tohoto algoritmu není zaručeno, že nalezne vždy nejkratší cestu, je ovšem výrazně rychlejší než Dijkstrův algoritmus díky použití heuristiky pro navigaci k cíli.



(a) Průběh hledání cesty bez překážky



(b) Průběh hledání cesty s konkávní překážkou

Obr. 4.2: Průběh hledání cesty pomocí Best-first algoritmu (zdroj: [9])

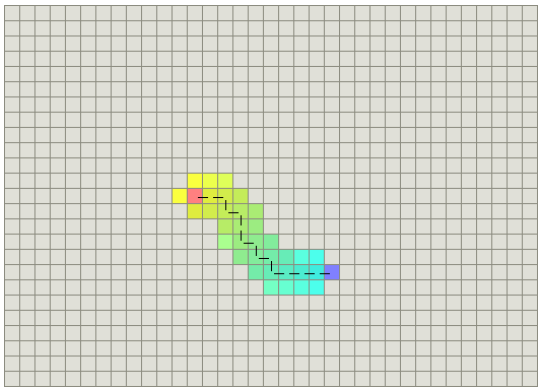
Na obrázku 4.2 je průběh prohledávání prostoru. Žlutá barva reprezentuje uzly s vysokou hodnotou heuristické funkce (vysoké náklady pro cestu do cíle) a černá uzly s nízkou hodnotou heuristické funkce (nízké náklady pro cestu do cíle). Na obrázku 4.2b jsou vidět problémy při hledání trasy s konkávní překážkou algoritmem Best-first, který se snaží co nejvíce přiblížit k cíli. Tyto problémy jsou způsobeny tím, že algoritmus uvažuje pouze náklady na vzdálenost k cíli a nezohledňuje délku trasy.

4.3 Algoritmus A*

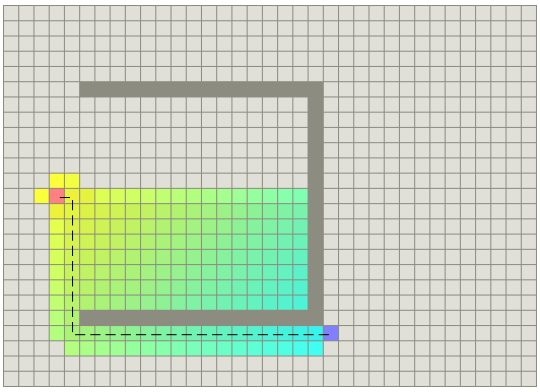
Algoritmus A* je nejčastěji používaným algoritmem pro hledání trasy pro jeho flexibilitu a široké možnosti použití [9]. Algoritmus může prohledávat velkou oblast mapy a hledat nejkratší cestu, stejně jako Dijkstrův algoritmus. Také využívá heuristiku k navigaci k cíli jako algoritmus Best-first. Pro jednoduché případy je stejně rychlý jako Best-first algoritmus.

A* algoritmus kombinuje vlastnosti Dijkstrova algoritmu (upřednostňuje body, které jsou blíže ke startu) a algoritmu Best-first (upřednostňuje uzly které jsou blíže k cíli). Funkce $g(n)$ reprezentuje náklady na délku trasy od startovního uzlu k uzlu n a funkce $h(n)$ reprezentuje heuristický odhad vzdálenosti uzlu n od cíle. Algoritmus zkoumá vždy uzel s nejnižší hodnotou $f(n) = g(n) + h(n)$.

Na obrázku 4.3 je průběh prohledávání algoritmem A*. Žlutá barva značí vzdálenost uzlu od cíle (hodnota heuristické funkce h) a modrá barva značí vzdálenost uzlu od startu (hodnota funkce g). Při hledání cesty s konkávní překážkou (obrázek 4.3b) prohledá algoritmus A* menší prostor než Dijkstrův a Best-first algoritmus a najde nejkratší cestu stejně jako Dijkstrův algoritmus.



(a) Průběh hledání cesty bez překážky



(b) Průběh hledání cesty s konkávní překážkou

Obr. 4.3: Průběh hledání cesty pomocí A* algoritmu (zdroj: [9])

4.4 Algoritmus Floyd-Warshall (FW)

Floydův-Warshallův algoritmus [1] porovnává všechny možné cesty v grafu mezi všemi dvojicemi jeho vrcholů. Algoritmus postupně vylepšuje odhad nejkratší cesty do doby, než je odhad optimální.

Graf obsahuje vrcholy $V = \{1, 2, \ldots, n\}$. Algoritmus vytvoří podmnožinu vrcholů $\{1, 2, \ldots, k\}$. Pomocí vrcholů z této množiny hledá nejkratší cestu mezi všemi vrcholy i a j z množiny V, jak ukazuje pseudokód 4.1. Jsou zde dvě možnosti: nejkratší cesta

Kód 4.1: Pseudokód algoritmu Floyd-Warshall

```
/* Predpokladame funkci edgeCost(i,j) vracejici cenu hrany z i do j
   (nekonecno pokud neexistuje). N je pocet vrcholu a
   edgeCost(i,i)=0 */
int path[][];
/* 2-rozmerna matice. V kazdem kroku algoritmu je path[i][j]
   nejkratsi cesta z i do j pouzitim 1. az k-teho vrcholu. Kazda
   path[i][j] je inicializovana na edgeCost(i,j). */
procedure FloydWarshall ()
   for k := 1 to n
      for i := 1 to n
         for j := 1 to n
            path[i][j] = min(path[i][j], path[i][k]+path[k][j]);
```

používá pouze vrcholy z podmnožiny $\{1, 2, \ldots, k\}$, nebo existuje cesta jdoucí z i do $k + 1$ a pak z $k + 1$ do j, která je kratší. Ve druhém případě spojíme cesty z i do

$k+1$ a z $k+1$ do j do nové nejkratší cesty. Algoritmus tento postup opakuje pro všechna $k \in \{1, 2, \ldots, n\}$.

5 IMPLEMENTACE NAVIGACE V TRASI

5.1 Topologická mapa

Topologická mapa reprezentovaná třídou `TopoMap` popisuje propojení jednotlivých křižovatek v mapě bez ohledu na jejich geometrické umístění. Každé spojení je hodnoceno váhou dané cesty. Váha je doba potřebná k projetí dané hrany a je vypočtena dle maximální povolené rychlosti dané hrany a její délky dle vzorce 5.1.

$$w = \frac{l \cdot 3{,}6}{v} \qquad [s] \tag{5.1}$$

kde:

l ..... délka hrany v metrech
v ..... maximální rychlost na dané hraně v km/h

Takovouto topologickou mapu je možné považovat za orientovaný graf, na základě kterého bude možné provést výpočet nejvýhodnější trasy mezi dvěma zadanými body (křižovatkami) pomocí algoritmů hledání cesty v orientovaných grafech.

5.2 Implementace algoritmů

Pro implementaci byly vybrány algoritmy A* a FW. Ve jmenném prostoru `Engine.Router` byla vytvořena abstraktní třída `BasicRouter`. Tato třída dědí z rozhraní `IRouter`. Obsahuje třídu `RouterCrossing` sloužící k uchovávání hodnoty funkce $f(n)$ (viz kapitola 4.3) pro danou křižovatku a také pomocné metody pro práci s topologickou mapou. Také obsahuje veřejné virtuální metody pro hledání trasy, které jsou rozděleny dle účelu takto:

Init Routing slouží pro prvotní nalezení trasy při vytvoření auta v simulaci.

Rerouting slouží k nalezení trasy, pokud se auto ztratilo.

Change Route nalezne novou trasu na základě nově zjištěných dopravních událostí.

Change Route Stochastic využívá principu stochastického routeru k nalezení trasy dle nově zjištěných událostí (viz kapitola 7).

Metody přijímají jako parametry objekt auta, počáteční a koncový bod trasy, poslední projetý bod ve formě objektu `Crossing` (křižovatka) a časové razítko vzniku požadavku. Metody `ReroutingRequest` a `ChangeRouteRequest` a `ChangeRouteStochasticRequest` navíc přijímají seznam dopravních událostí, které se berou v potaz při hledání trasy (více o dopravních událostech v kapitole 6). Metoda `ChangeRouteStochasticRequest` ještě navíc přijímá starou trasu auta.

Tyto metody zařadí příchozí požadavek do fronty, kterou následně zpracovává vlákno routeru (viz kapitola 5.3). Výstupem routeru po zpracování daného požadavku je seznam křižovatek, které musí auto projet. Router používá privátní virtuální metodu `Routing`, která realizuje samotné hledání trasy a je pro každý vyhledávací algoritmus jiná.

Dále třída obsahuje statické proměnné a metody pro uchovávání statistik o počtu provedených hledání trasy. Mezi sledované statistiky patří počet prvotních hledání trasy (proměnná `InitRoutingCount`), které se uskuteční pro každé auto pouze jednou při jeho vzniku. Dále počet hledání alternativních tras (proměnná `ReroutingCount`), ke kterým dochází pokud se auto dostane na silnici, která nesměřuje do požadované křižovatky dle trasy. Poslední sledovanou statistikou je počet změn trasy na základě změny dopravní situace (viz kapitola 6), která je uchovávána v proměnné `ChangeRoutingCount`.

5.3 Zpracování požadavků

Zpracování požadavků probíhá paralelně ve více vláknech bez ohledu na typ routovacího algoritmu. Třída `BasicRouter` obsahuje instanci třídy `RouteThreadHelper`, která uchovává FIFO frontu požadavků a objekt pro práci s vlákny, které požadavky zpracovávají. Příchozí požadavky na nalezení trasy jsou zpracovány výše popsanými metodami a jsou zařazeny do fronty požadavků. Algoritmus routerů běží v samostatných vláknech (počet vláken je zvolen podle počtu jader procesoru) a z této fronty postupně zpracovávají požadavky. Po vyřízení požadavku je nová trasa zaslána žádajícímu autu, které aktualizuje směr své jízdy.

Pokud je ve frontě požadavek starší než nastavená hodnota (výchozí hodnota je 10 s a je ji možné změnit v nastavení simulátoru), přejde router do stavu přetížení. V takovém případě zašle požadavek vláknu, ve kterém běží samotná simulace, na pozastavení jejího běhu. Jakmile jsou všechny požadavky ve frontě zpracovány, opět je simulační vlákno informováno, že je možné pokračovat v simulaci.

5.4 Implementace algoritmu A*

Algoritmus A* je implementován ve třídě `ASrouter` ve jmenném prostoru `Engine.Router`. Tato třída dědí ze základní třídy `BasicRouter` popsané výše.

5.4.1 Princip fungování algoritmu

Algoritmus pracuje s topologickou mapou (viz kapitola 5.1), kde každý uzel má svůj jedinečný index. Algoritmus si uchovává dva seznamy prošlých bodů. První uchovává

seznam otevřených uzlů (`OpenSet`), mezi kterými vyhledává při každé iteraci ten s nejnižší hodnotou funkce $f(n)$ (viz kapitola 4.3). Druhý uchovává seznam uzlů, které již algoritmus zahrnul do trasy a které už nejsou při dalším výpočtu uvažovány. Dále algoritmus uchovává pole `ComeFrom`, v němž jsou uchovány indexy uzlů, ze kterých vede trasa do uzlu určeného indexem v tomto poli. Pole `Gscore`, `Hscore` a `Fscore` slouží k uchování výsledků funkcí $g(n)$, $h(n)$ a $f(n)$, popsaných v kapitole 4.3, pro každý uzel.

Algoritmus provádí tyto kroky:

1. Vypočítá hodnotu funkce $f(n)$ pro počáteční uzel, do pole `ComeFrom` na pozici určenou indexem daného uzlu zapíše hodnotu -1 a vloží tento uzel do seznamu otevřených uzlů.
2. Vyhledá uzel s nejnižší hodnotou funkce $f(n)$.
3. Pokud je nalezený uzel cílový, hledání končí.
4. Umístí tento uzel do seznamu uzavřených uzlů a odstraní jej ze seznamu otevřených uzlů.
5. Pomocí topologické mapy projde všechny uzly propojené s uzlem nalezeným v bodě 2.
6. Pro každý propojený uzel provede jednu z následujících operací:
   - Pokud se daný uzel nachází v seznamu uzavřených uzlů, pokračuje na další uzel.
   - Pokud se uzel nenachází v seznamu otevřených uzlů, vypočte hodnoty funkcí $g(n)$, $h(n)$ a $f(n)$ pro daný uzel, přidá ho do seznamu otevřených uzlů a zaznamená si index uzlu, získaného v bodě 2, do pole `ComeFrom` na pozici, určenou indexem aktuálního zkoumaného uzlu.
   - Pokud je uzel v seznamu otevřených uzlů a zároveň je aktuální hodnota funkce $g(n)$ nižší, než hodnota uložená v poli `Gscore` pro daný uzel, aktualizuje pole `ComeFrom` aby obsahovalo pro daný uzel index uzlu, získaného v bodě 2. Poté aktualizuje hodnotu v poli `Gscore` a vypočítá novou hodnotu funkce $f(n)$.
7. Algoritmus pokračuje bodem 2.

Po skončení hledání je zavolána metoda `FindPath`, která přijímá jako parametry pole `ComeFrom` a index cílového bodu. Metoda prochází pole `ComeFrom` a dle indexů v něm uložených vyhledává a ukládá uzly (křižovatky) do seznamu, jak ukazuje algoritmus 5.1. Jakmile nalezne v poli hodnotu -1 (počáteční uzel trasy) prohledávání končí, je obráceno pořadí v seznamu křižovatek (pole se prohledávalo od cíle) a tento seznam je vrácen jako výstup metody.

Kód 5.1: Metoda FindPath pro rekonstrukci trasy algoritmu A*

```
private static List<Crossing> FindPath(ref int[] ComeFrom,
    int index)
{
    List<Crossing> list = new List<Crossing>();

    while (ComeFrom[index] >= 0)
    {
        list.Add(Infrastructure.TopoMapa.Crossings[index].
            ThisCrossing);
        index = ComeFrom[index];
    }
    list.Reverse();
    return list;
}
```

5.5 Implementace algoritmu FW

Algoritmus FW je implementován ve třídě `FloydWarshallRouter` ve jmenném prostoru `Engine.Router`. Tato třída dědí ze základní třídy `BasicRouter` popsané výše. Obsahuje stejné veřejné metody pro hledání trasy jako třída `ASrouter`. Jedinou odlišností jsou metody `Routing` a `FindPath`.

5.5.1 Princip a fungování algoritmu

Algoritmus si uchovává dvě dvourozměrné pole. První pole `path` uchovává váhu pro každý pár uzlů. Váha je spočítána pomocí metody `EdgeCost`, která vrací nulu pro shodné prvky, váhu z topologické mapy pro pár uzlů, mezi kterými existuje přímé spojení a nekonečno (maximální hodnotu daného datového typu) pro pár uzlů, mezi kterými neexistuje přímé spojení. Druhé pole `next` slouží pro rekonstrukci trasy.

Algoritmus nejdříve provede inicializaci polí a poté provede hledání trasy jak ukazuje algoritmus 5.2. Rekonstrukce trasy potom probíhá rekurzivně pomocí metody `GetPath`, jak ukazuje algoritmus 5.3.

Kód 5.2: Vyhledávání trasy pomocí algoritmu FW

```
for (int k = 0; k < n; k++)      // searching path
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (path[i, k] + path[k, j] < path[i, j])
            {
                path[i, j] = path[i, k] + path[k, j];
                next[i, j] = k;
            }}}}
```

Kód 5.3: Rekonstrukce trasy pomocí algoritmu FW

```
private static bool GetPath(int i, int j, ref List<Crossing>
   route, double[,] path, int[,] next, List<Junction> TopoMapa,
    int recurse)
{
    recurse++;
    if (recurse > 500)
        return false;     // unable to find path
    if (path[i, j] == double.MaxValue)
        return false;     // unable to find path
    int predchozi = next[i, j];
    if (predchozi == j)
        return false;     // unable to find path
    if (predchozi == -1)
        return true;          /* there is an edge from i to j,
           with no vertices between */
    if (!GetPath(i, predchozi, ref route, path, next, TopoMapa,
       recurse))
        return false;     // unable to find path
    route.Add(TopoMapa[predchozi].ThisCrossing);
    if (!GetPath(predchozi, j, ref route, path, next, TopoMapa,
       recurse))
        return false;     // unable to find path
    return true;
}
```

6 UDÁLOSTI A KOMUNIKAČNÍ VRSTVA

Události slouží k hlášení dopravní situace. Základní typy dopravních situací jsou:

- Stání na místě (`LongStanding`)
- Nízká rychlost (`LowSpeed`)
- Dopravní hlášení (`TrafficReport`)

Pokud se auto dostane do dané dopravní situace, vytvoří příslušnou událost. Událost obsahuje unikátní identifikátor vozidla, místo vzniku události, čas vzniku události, dobu platnosti, poloměr platnosti zprávy, penalizaci a údaj specifický pro danou událost. Místo události je specifikováno dvěma křižovatkami, mezi kterými k události došlo, silnicí (`Edge`), místem kde k události došlo a vzdáleností od poslední projeté křižovatky. Tatko vzniklá událost je předána komunikační vrstvě, jejíž úkolem je předat tuto událost dalším vozidlům. Ostatní vozidla pak mohou na základě těchto událostí měnit svoji trasu tak, aby se vyhnula případným dopravním komplikacím. Podrobněji se problematice komunikační vrstvy věnuje kapitola 6.3.

6.1 Typy událostí

6.1.1 Událost stání na místě

Událost typu `LongStanding` (stání na místě) slouží k hlášení stání auta (například uváznutí v dopravní zácpě apod.). Pokud auto jede pomaleji než $1{,}3\,m \cdot s^{-1}$ déle než nastavenou hodnotu simulačního času (výchozí hodnota je 5 s) a je od nejbližší křižovatky dál než čtyřnásobek jeho délky, vytvoří událost typu `LongStanding`. Zpráva obsahuje čas stání v sekundách. Výpočet penalizace probíhá následovně:

$$s_{penalty} = t_{standing} \cdot v_{max}$$

6.1.2 Událost nízká rychlost

Událost typu `LowSpeed` (nízká rychlost) slouží k hlášení snížené rychlosti v daném úseku (například snížená průjezdnost vlivem vysoké hustoty provozu – vznik kolon). Podmínkou pro vznik události je, že auto jede rychleji než $1{,}3\,m \cdot s^{-1}$, ale pomaleji než 80 % maximální rychlosti vozidla a zároveň pomaleji než 80 % maximální povolené rychlosti na silnici. Pokud je tato podmínka splněna déle než nastavenou hodnotu simulačního času (výchozí hodnota je 5 s) a je od nejbližší křižovatky dál než čtyřnásobek jeho délky, auto vytvoří zprávu typu `LowSpeed`. Tato zpráva obsahuje údaj o době jízdy po dané silnici, ujetou vzdálenost a maximální možnou rychlost na

daném úseku (případně maximální rychlost auta, pokud je povolená rychlost vyšší než maximální rychlost auta). Výpočet penalizace probíhá takto:

$$s_{penalty} = s_{route} \cdot \left(\frac{v_{max}}{v_{actual}} - 1 \right)$$

6.1.3 Událost dopravní hlášení

Událost typu `TrafficReport` (Dopravní hlášení) slouží k hlášení snížené průjezdnosti daného úseku. Na rozdíl od události `LowSpeed` zahrnuje i stání před křižovatkou. Toto hlášení probíhá až při opuštění dané cesty (`Edge`) v případě, že čas průjezdu byl o nastavenou hodnotu (výchozí je 50 %) delší než předpokládaná doba jízdy (vypočtená z délky cesty a její maximální povolené rychlosti nebo maximální rychlosti auta). Zpráva obsahuje nadbytečný čas, potřebný k ujetí dané cesty. Penalizace je vypočítána dle tohoto vztahu:

$$s_{penalty} = t_{extra} \cdot v_{max}$$

6.2 Implementace událostí

Události jsou definované ve jmenném prostoru `Engine.TrafficEvents`. Základní událostí je `BasicTrafficEvent`, ze které ostatní události dědí. Obsahuje základní data společná pro všechny události, jak ukazuje kód 6.1. Obsahuje také třídu `EventLocation`, která uchovává informace o pozici události (kód 6.2 ukazuje její proměnné). Proměnná `Penalty` obsahuje penalizaci hrany na základě dat z události.

Kód 6.1: Proměnné ve třídě `BasicTrafficEvent`

```
public Int64 ID {get; private set;}      //Unique ID
public readonly EventLocation Location; // Location of event
public TimeSpan Timestamp {get; private set;}   // Time of Event
public double TTL;       // Time to live
public static int Radius;        // Area of validity in meters
// Type of event
public EventType Type {protected set; get;}
//Penalty for the edge from event's data
public double Penalty {protected set; get;}
```

Penalizace je následně přičtena k ceně hrany v topologické mapě při hledání trasy. Třída také obsahuje virtuální metody pro uchovávání statistik o používání událostí.

Kód 6.2: Část třídy `EventLocation`

```
public sealed class EventLocation
{
    public Crossing From, To;
    public Edge Edge;
    public double Distance;
    public PointF Location;
}
```

Ostatní události se nacházejí ve stejném jmenném prostoru a dědí z třídy `BasicTrafficEvent`. Každá událost má svůj vlastní konstruktor, který navíc přijímá data specifická pro danou událost. Každá událost má také vlastní algoritmus výpočtu penalizace.

6.3 Typy komunikačních vrstev

V simulátoru jsou dva typy komunikační vrstvy:

- Car to Car (C2C)
- Car to Infrastructure (C2I)

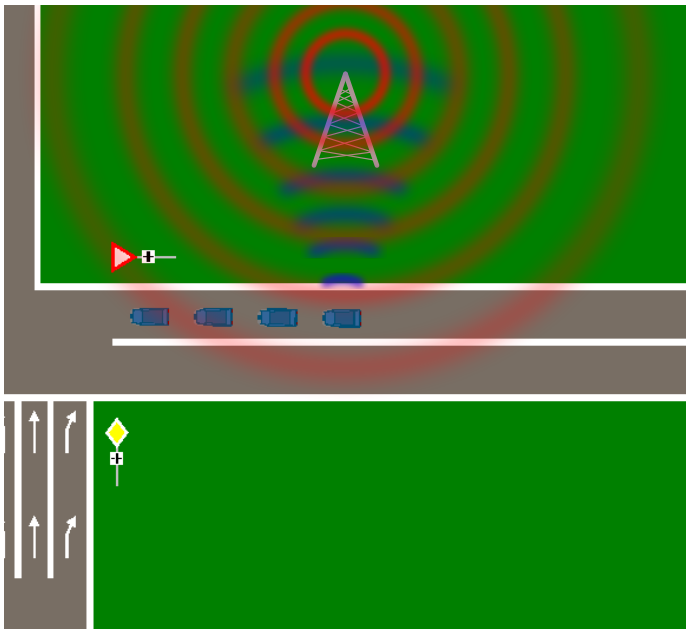
6.3.1 Car to Infrastructure (C2I)

Tato komunikační vrstva simuluje předávání zpráv pomocí infrastruktury. Infrastruktura je tvořena sítí přijímačů a vysílačů, které zachytávají a distribuují bezdrátově šířené zprávy od vozidel (viz obrázek 6.1). Infrastruktura má přehled o všech dopravních událostech a všechny události jsou distribuovány po celé oblasti. Zprávy se tedy šíří do celé oblastí zároveň a není nutné si je postupně předávat.

Výhody této vrstvy jsou, že není potřeba velkého počtu aut v dané oblasti schopných tímto způsobem komunikovat a rychlost a spolehlivost šíření zpráv. Nevýhodou je nutnost vybudování infrastruktury v daném místě (například ve městě) a její údržba.

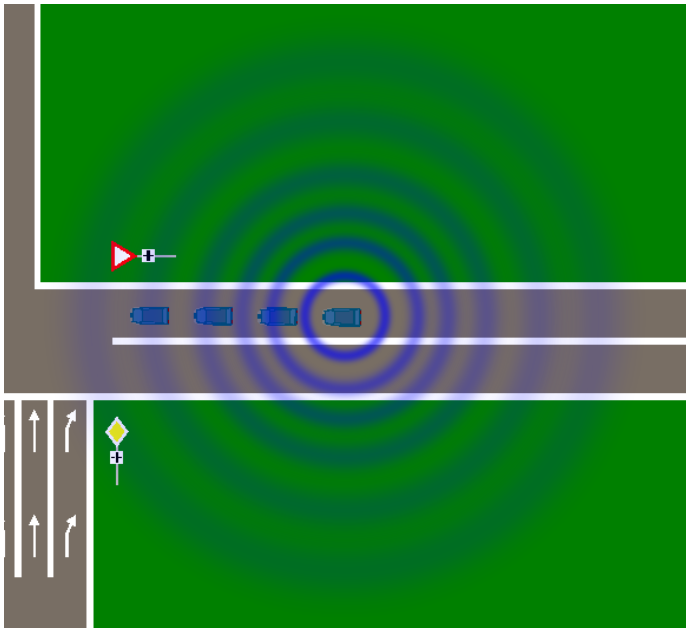
6.3.2 Car to Car (C2C)

Tato komunikační vrstva pracuje tak, že simuluje přímé bezdrátové předávání zpráv mezi jednotlivými auty. Auto „vysílá“ události do svého okolí a všechny auta v jeho dosahu zprávu přijmou (jak ukazuje obrázek 6.2) a následně samy tuto zprávu, spolu



Obr. 6.1: Komunikace mezi auty prostřednictvím vrstvy C2I

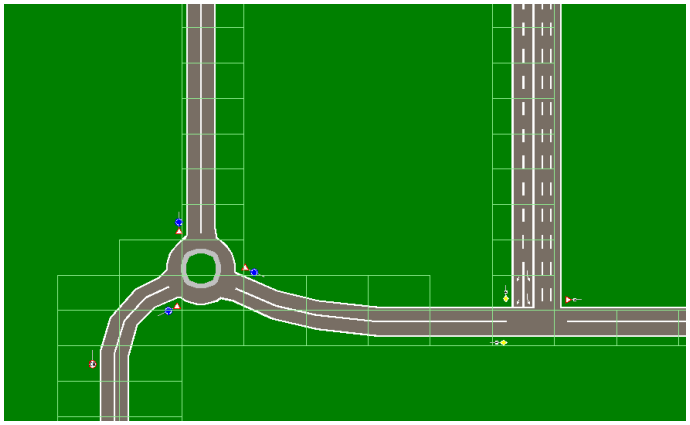
se svými vlastními, stejným způsobem odesílají dále. Zpráva se tedy v dané oblasti šíří postupně přes jednotlivé auta, schopné tímto způsobem komunikovat.



Obr. 6.2: Komunikace mezi auty prostřednictvím vrstvy C2C

Tato vrstva je z hlediska snadnější implementace navržena tak, že se na dané mapě vytvoří tzv. komunikační grid. Každá buňka gridu obsahuje komunikační vrstvu typu C2I a seznam sousedních buněk. Při odesílání zprávy do jedné buňky gridu jsou

zprávy automaticky předány sousedním buňkám a to v nastavené vzdálenosti, čímž se simuluje dosah vysílače v automobilu. Z hlediska optimalizace grid pokrývá pouze úseky, kde se nachází silnice, jak ukazuje obrázek 6.3.



Obr. 6.3: Komunikační grid ve vrstvě C2C (zelené obdélníky)

Výhody tohoto způsobu komunikace jsou nulové náklady na vytvoření a údržbu komunikační sítě. Nevýhody jsou nutnost dostatečného počtu aut schopných komunikovat tímto způsobem v dané oblasti a rychlost a spolehlivost šíření zpráv. V případě, že v dané oblasti není dostatek aut, zprávy se nešíří a ostatní auta se o dopravní situaci nedozví.

6.4 Implementace komunikačních vrstev v TRASI

Základní infrastruktura je implementována ve jmenném prostoru `SimCity.Engine.EventLayers` v abstraktní třídě `BasicEventLayer`. Tato třída dědí z rozhraní `IEventLayer` a obsahuje seznam zpráv `Events`, seznam nově přijatých zpráv `NewEvents` a základní metody pro příjem a odesílání zpráv.

Pro čtení zpráv z komunikační vrstvy (příjem zpráv) slouží metoda `RecieveEvent`. Tato metoda přijímá pozici dotazujícího se auta a vrací seznam pro něj platných zpráv (vzhledem k oblasti platnosti zprávy) ze seznamu již dříve přijatých zpráv (`Events`).

Odesílání zpráv je implementováno v metodách `SendEvent` a `SendEvents`, které přijímají událost, respektive seznam událostí a pozici odesílajícího auta. Nově přijaté zprávy jsou testovány na duplicitu podle silnice (`Edge`), na které událost vznikla. Zprávy, které pocházejí ze stejné silnice jako již dříve přijatá zpráva (v seznamu `NewEvents`) od jiného auta, jsou testovány na velikost penalizace a shodnost směru jízdy (ze které křižovatky auto jelo a kam směřovalo). Pokud se směr jízdy neshoduje s žádnou již přijatou zprávou, je tato zpráva přidána jako nová unikátní zpráva.

Pokud se směr shoduje a nově přijatá zpráva má vyšší penalizaci než zpráva již dříve přijatá, je starší zpráva nahrazena novou.

Třída také obsahuje metodu `UpdateLayer`, která je volána po každém simulačním kroku a jejím úkolem je smazat všechny události v seznamu `Events` a přesunout do něj nově přijaté události ze seznamu `NewEvents`.

6.4.1 Implementace vrstvy C2I

Tato vrstva je implementována ve třídě `C2ILayer`. Neobsahuje žádné vlastní definice, pouze dědí ze třídy `BasicEventLayer`.

6.4.2 Implementace vrstvy C2C

Vrstva je implementována ve třídě `C2CLayer`. Tato třída dědí z rozhraní `IEventLayer` a obsahuje třídy `gridParameters`, `Cell` a samotný komunikační grid v proměnné `Grid`.

Komunikační grid je typu `Dictionary` a obsahuje objekty typu `Cell`, což jsou komunikační vrstvy typu C2I obsahující navíc pozici a seznam sousedních buněk.

Metody pro příjem a odesílání jsou upraveny tak, aby volaly příslušné metody v dané buňce gridu (kód 6.3 ukazuje metodu `SendEvents`).

Kód 6.3: Metoda `SendEvents` ve třídě `C2CLayer`

```
public void SendEvents(List<BasicTrafficEvent> events, PointF
    location)
{
    // is car in grid?
    if ((location.X < GridParameters.Position.X) || (location.Y <
         GridParameters.Position.Y) ||
        (location.X >= (GridParameters.Position.X + GridParameters
            .Width)) || (location.Y >= (GridParameters.Position.Y +
             GridParameters.Height)))
          return;

    int ID = GetGridID(location);
    if (!Grid.ContainsKey(ID)) return;

    Grid[ID].SendEvents(events, location);

    // send event to neighbouring cells
    foreach (Cell cell in Grid[ID].Neighbors)
    {
        cell.SendEvents(events, location);
    }
}
```

7 STOCHASTICKÁ NAVIGACE

Úkolem stochastické navigace je snížit zátěž objízdné trasy. Při vzniku dopravní komplikace v případě klasické navigace obdrží všechny auta, projíždějící danou oblastí, stejnou objízdnou trasu. Tato objízdná trasa ovšem nemusí mít dostatečnou kapacitu, aby dokázala vzniklý provoz pojmout.

Stochastický router místo toho navrhne několik alternativních objízdných tras a na základě jejich dopravní propustnosti náhodně (stochasticky) zvolí jednu z těchto tras. Pravděpodobnost výběru trasy je větší pro trasy s vyšší propustností. Důsledkem je, že dvě vozidla na prakticky stejném místě ve stejném okamžiku se stejným cílovým bodem mohou od stejné navigace dostat odlišnou náhradní trasu. Tímto chování by mělo být docíleno rovnoměrného rozložení dopravní zátěže na více objízdných tras a tím celkově vyšší propustnosti a plynulosti provozu v dané oblasti.

7.1 Topologická mapa propojení

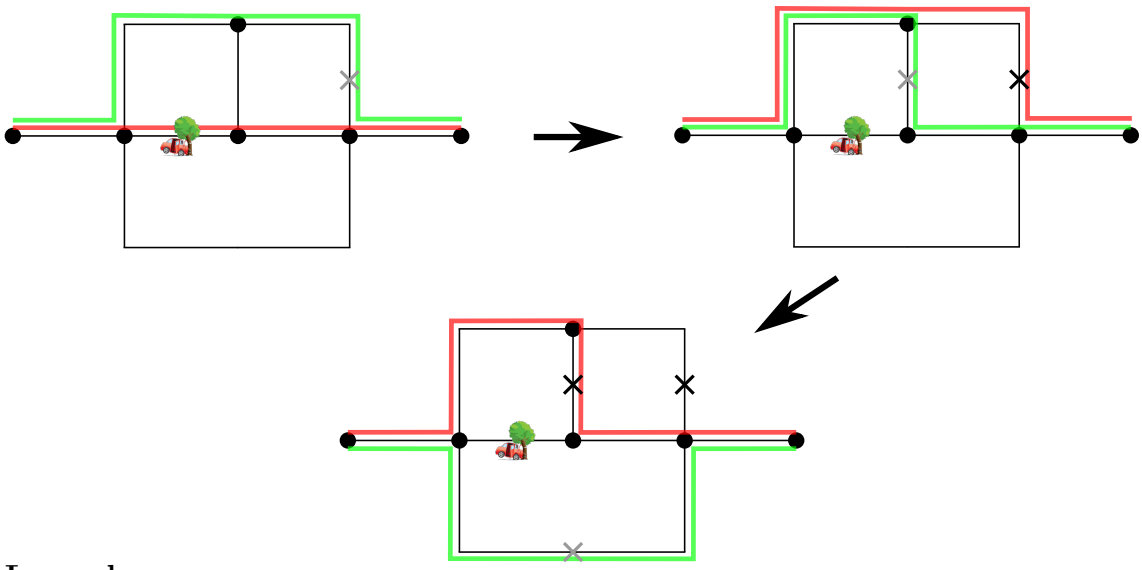
Pro účely stochastického routeru byla vytvořena topologická mapa propojení (`ConnectionMap`). Tato mapa uchovává počet propojení (počet pruhů) a minimální rychlost toho spojení mezi jednotlivými sousedícími křižovatkami. Pokud je mezi křižovatkami více pruhů, je minimální rychlost násobena jejich počtem pro zohlednění vyšší propustnosti dané cesty. Na základě těchto údajů určuje stochastický router pravděpodobnost, s jakou vybere některou z alternativních tras.

7.2 Princip a implementace

Stochastický router je implementován jako metoda `ChangeRouteStochastic` ve třídě `BasicRouter` (viz kapitola 5). Stejně jako ostatní metody pro hledání trasy využívá virtuální metodu `Routing`, která je odlišná pro každý vyhledávací algoritmus. Oproti ostatním typům routování ovšem hledá více alternativních tras.

Aby bylo možné najít alternativní trasu, využívá algoritmus znalosti původní trasy auta. Nejdříve najde alternativní cestu na základě hlášení o dopravní situaci a nově nalezenou trasu porovná s původní trasou auta. Poté postupuje po jednotlivých křižovatkách ve směru od cíle a hledá místo, kde se trasy liší. Jakmile toto místo nalezne, upraví topologickou mapu tak, že odstraní spojení mezi první odlišnou křižovatkou (oproti staré trase) a poslední shodnou křižovatkou. Poté opět zavolá router, který tím pádem nalezne další alternativní trasu. Dříve nalezená alternativní trasa je prohlášena za původní trasu auta a provede se opět hledání rozdílu oproti

nově nalezené alternativní trase. Tento postup je opakován dokud není nalezeno požadované množství alternativních tras nebo dokud již router nemůže nalézt žádnou další trasu. Postup hledání alternativních tras je na obrázku 7.1.



Legenda:

- Křižovatka
- Silnice
- Nově navržená trasa
- Stará trasa
- Dopravní událost
- Zrušené propojení
- Propojení, které bude zrušeno v dalším kroku

Obr. 7.1: Posloupnost hledání alternativních tras stochastickým routerem

Po nalezení všech tras je provedeno vyhodnocení vah pro každou cestu. Každá trasa je procházena po jednotlivých dvojicích křižovatek a z topologické mapy spojení (viz kapitola 7.1) je získávána rychlost na daném propojení. Pro každou trasu je zjištěna vždy nejnižší hodnota rychlosti, která se použije pro určení váhy této trasy.

Výběr trasy je pak proveden metodou vážené rulety. Váženou ruletu si lze představit jako klasickou ruletu s jedním rozdílem. Členění klasické rulety je rovnoměrné a vymezuje každému možnému losovanému číslu kruhovou výseč o stejném úhlu. U vážené rulety je pravděpodobnost výběru a tedy i velikost kruhové výseče určena váhou dané trasy [4].

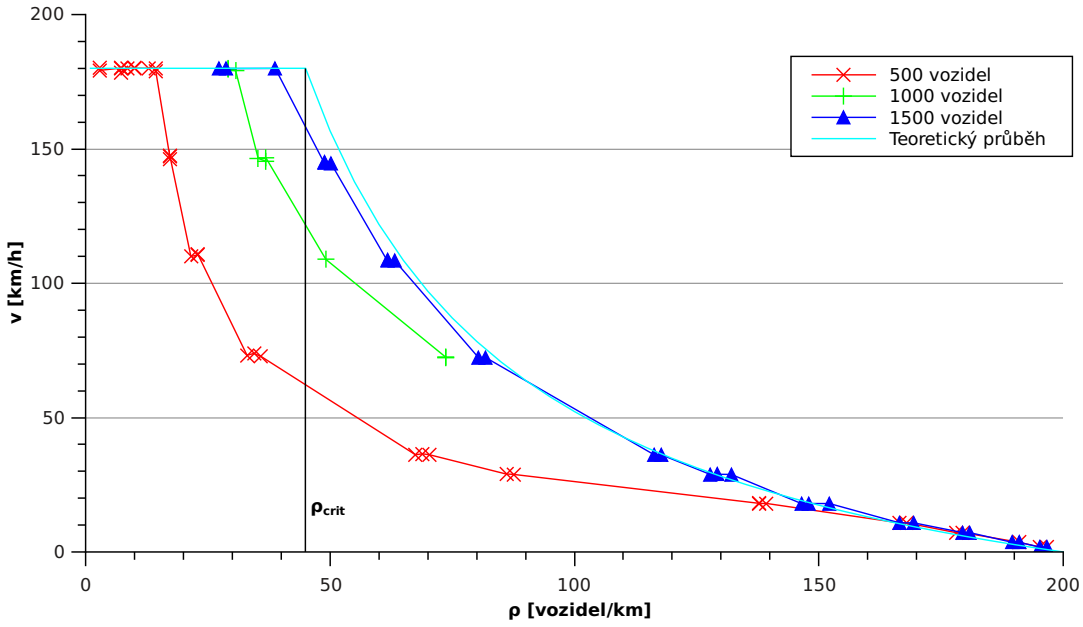
8 TESTY

8.1 Ověření simulátoru TRASI

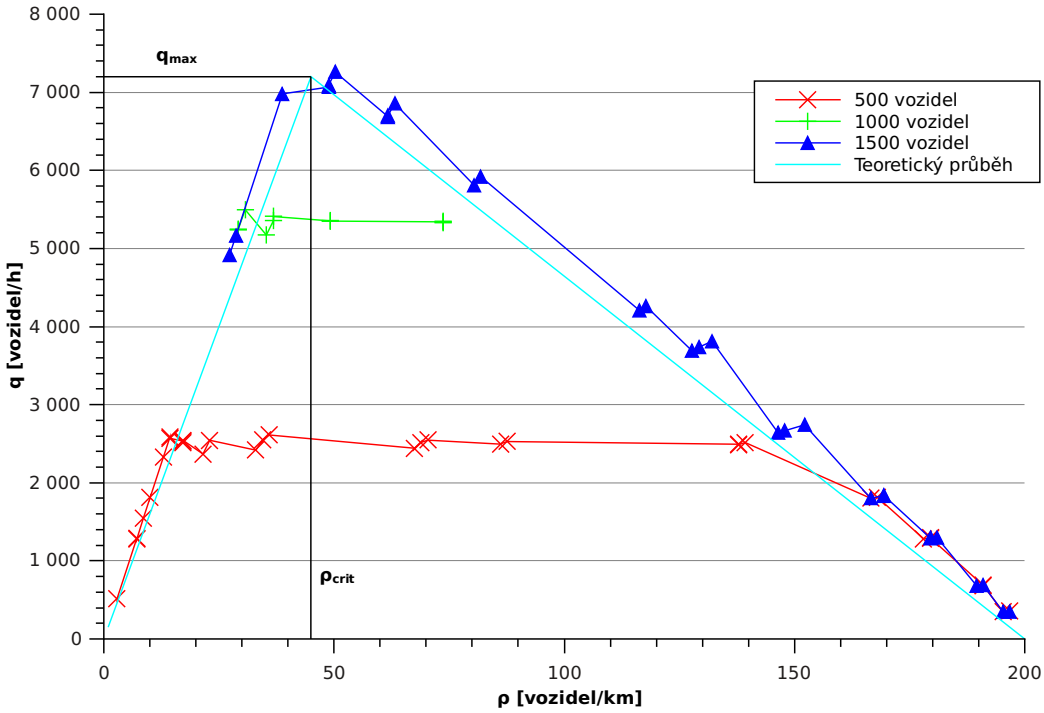
Pro ověření správnosti implementace dopravního modelu v simulátoru TRASI byl použit experiment se simulací části dálnice. Na obrázku 8.1 je vztah mezi rychlostí dopravního toku a hustotou. Na obrázku jsou různé průběhy, které odpovídají různému počtu vozidel vytvořených za dobu simulace. Poslední průběh ukazuje teoretické hodnoty vypočtené dle vztahu (3.14).

Vztah mezi dopravním tokem a hustotou je na obrázku 8.2. Průběhy jsou stejné jako u předchozího obrázku s tím rozdílem, že teoretické hodnoty byly vypočteny dle vztahu (3.16).

Vzhledem k tomu, že naměřené průběhy odpovídají teoretickým předpokladům, je možné tvrdit, že je dopravní model v simulátoru TRASI implementován správně. Podrobnější informace o testech simulátoru je možné nalézt v [5, 6].



Obr. 8.1: Závislost rychlosti na hustotě dopravy pro různé množství vozidel



Obr. 8.2: Závislost dopravního toku na hustotě dopravy pro různé množství vozidel

8.2 Metodika testování navigačních algoritmů

Pro účely testování bylo vytvořeno pět map (viz obrázek 8.3). První čtyři mapy byly uměle vytvořeny, poslední mapa (obrázek 8.3e) byla importována z OpenStreetMap[1] a jedná se o centrum města Hlinsko (více o importu mapových podkladů v [8]). Na každé mapě je jedno stojící vozidlo (na obrázcích označeno modrým kolečkem), které blokuje provoz v jednom jízdním pruhu. Toto vozidlo simuluje dopravní problém.

Pro každou mapu bylo provedeno 20 spuštění simulace. Pro komunikaci mezi vozidly byla využita komunikační vrstva C2I a vozidla využívala routovací algoritmus A*. V simulaci byly sledovány a vyhodnocovány tyto parametry:

Množství stojících vozidel bylo zaznamenáváno jako globální statistika každých 5 s běhu simulace. Udává průměrný počet vozidel, které měly v době měření nižší rychlost než $1\,m \cdot s^{-1}$.

Doba průjezdu 1000 vozidel je průměrná doba od příjezdu prvního vozidla (při zpracování výsledků se prvních 150 vozidel ignorovalo) do svého cíle po příjezd tisícého vozidla do svého cíle.

[1] Dostupné na adrese `www.openstreetmap.org`

Doba stání je průměrná doba stání vozidel. Za stání se považuje stav, kdy vozidlo jede nižší rychlostí než $1{,}3\,m \cdot s^{-1}$.

Doba pomalé jízdy je průměrná doba, po kterou vozidlo jelo rychleji než $1{,}3\,m \cdot s^{-1}$, ale pomaleji než 80 % maximální rychlosti vozidla a zároveň pomaleji než 80 % maximální povolené rychlosti na silnici.

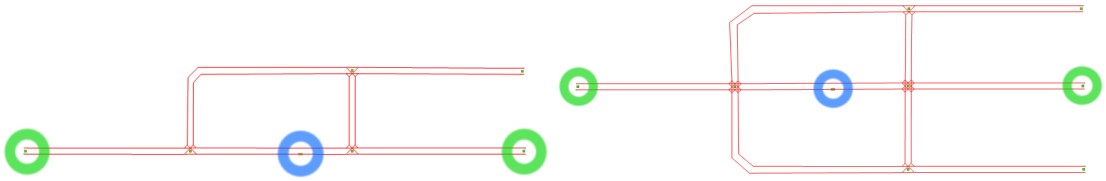
Ujetá vzdálenost je průměrná vzdálenost, kterou vozidla při průjezdu mapou ujedou.

Doba jízdy vozidel je průměrná doba, kterou vozidla stráví v simulaci než dosáhne svého cílového bodu.

První parametr byl zaznamenáván jako globální statistika každých 5 s běhu simulace. Ostatní parametry se zaznamenávaly individuálně pro každé vozidlo ve chvíli, kdy dorazilo do cíle. Výstupem každé simulace jsou soubory se záznamem globálních a individuálních statistik. Tyto soubory byly zpracovány tak, že z každé simulace byla vypočítána průměrná hodnota každého sledovaného parametru. Z výsledných hodnot byly vykresleny krabicové grafy, viz následující kapitoly.

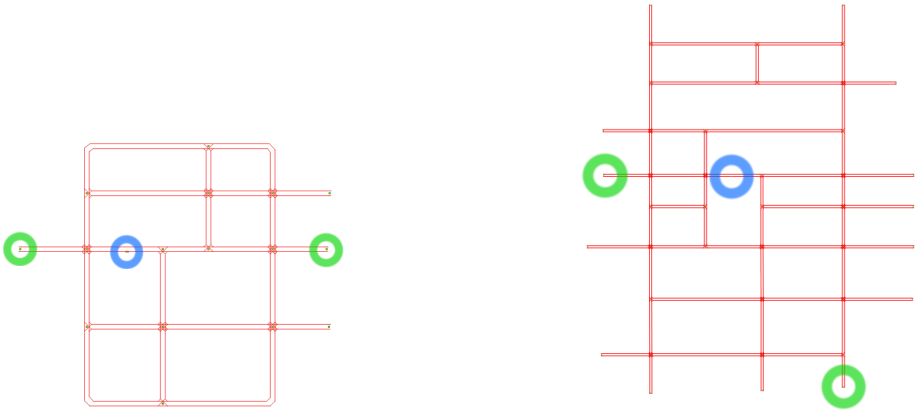
Při spuštění simulace je mapa prázdná a vozidla jsou náhodně generována z uzlů zvaných *CarGen* a jejich cíl cesty je vybírán náhodně ze všech cílových uzlů *RoadEnd* dle jeho atraktivity (pravděpodobnosti že bude daný cíl vybrán). Na každé mapě jsou dva cílové uzly s vyšší atraktivitou a větším množstvím vygenerovaných vozidel než ostatní (označeny zeleným kolečkem), což představuje hlavní silnici s vyšším provozem. Množství generovaných vozidel je u hlavní silnice voleno u většiny map $2\times$ až $3\times$ vyšší než u vedlejších silnic. Poměr byl u každé mapy zvolen experimentálně tak, aby nedocházelo k naprostému zablokování dopravy ale zároveň byla doprava dostatečně hustá k ověření vlivu typu navigace. Tento stav simuluje značně zatíženou dopravní infrastrukturu města.

První projíždějící vozidla mají volnou cestu a data z nich nejsou relevantní. Proto bylo ze souboru globálních statistik ignorováno prvních 60 s simulace a ze souboru individuálních statistik bylo ignorováno prvních 150 vozidel.



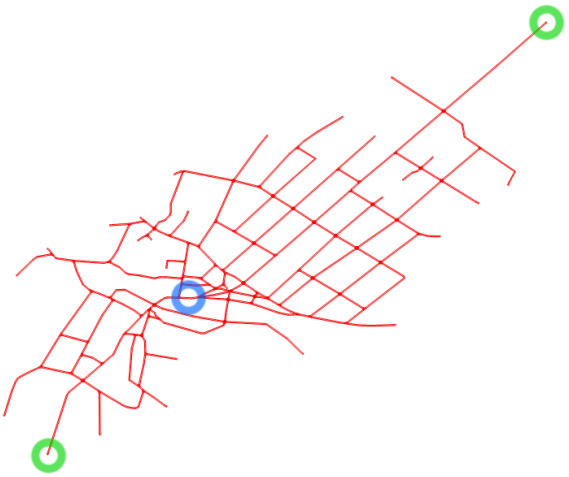
(a) Test 1

(b) Test 2



(c) Test 3

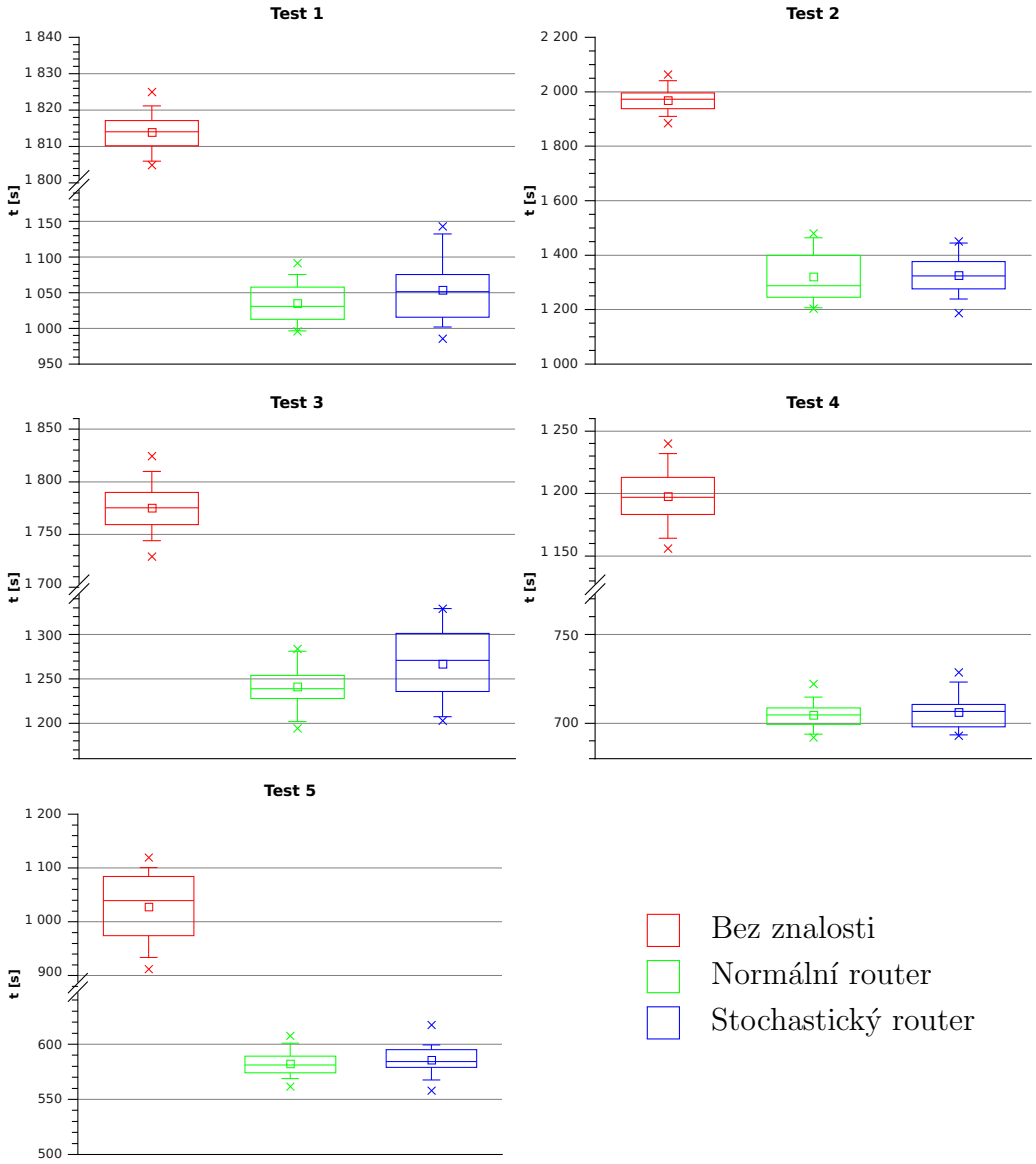
(d) Test 4



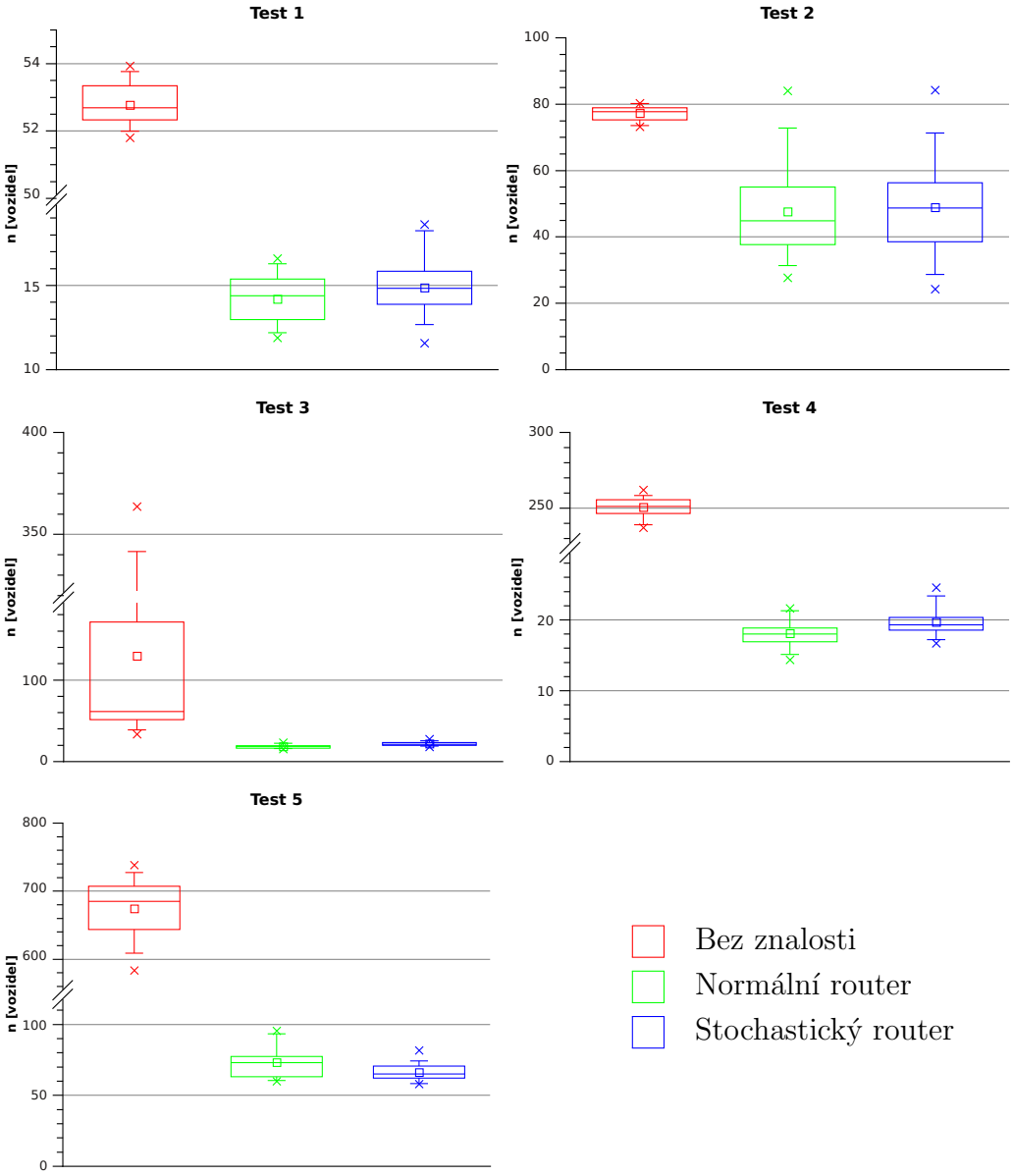
(e) Test 5 – Hlinsko

Obr. 8.3: Testované mapy

8.3 Porovnání metod navigace

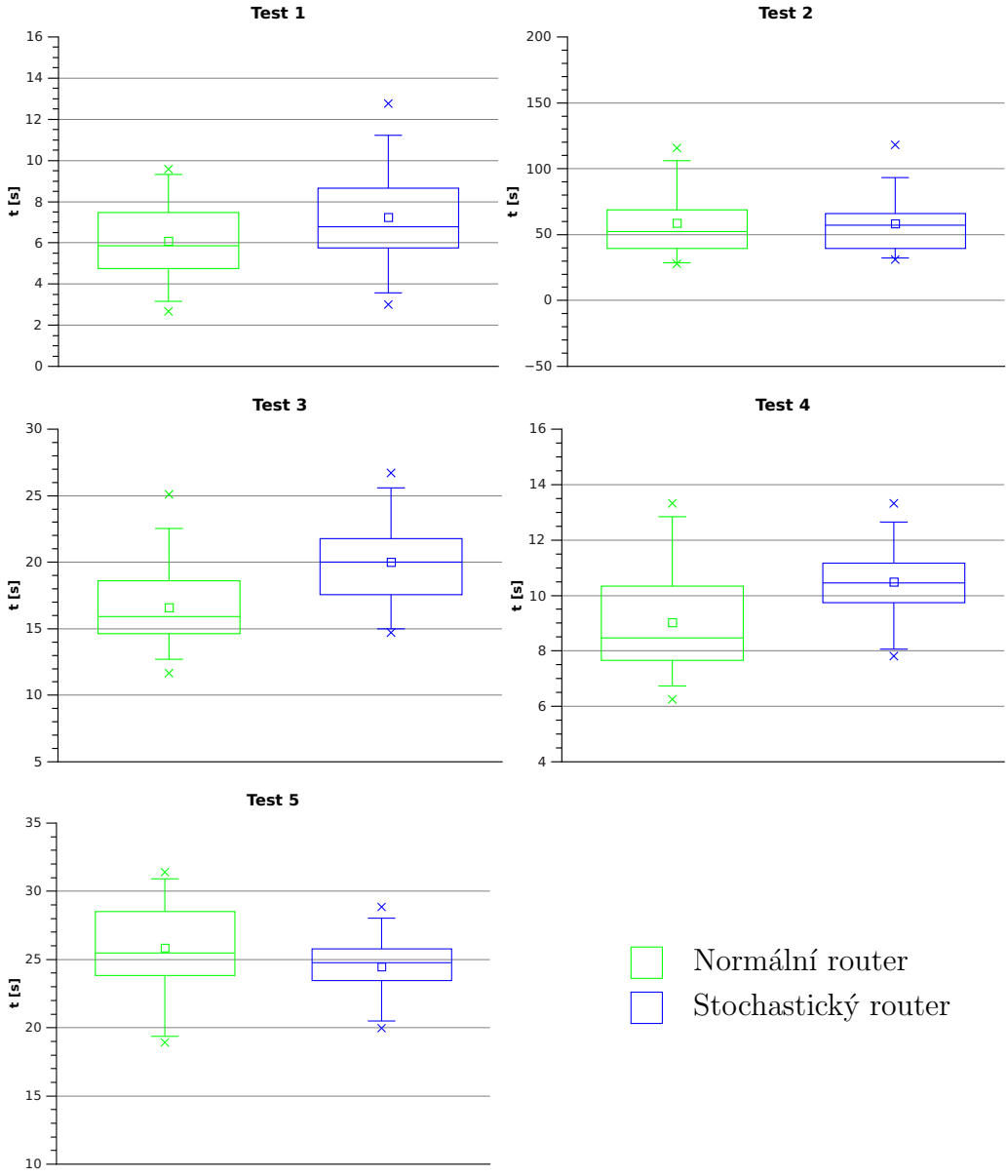


Obr. 8.4: Závislost doby průjezdu 1000 vozidel na typu navigace

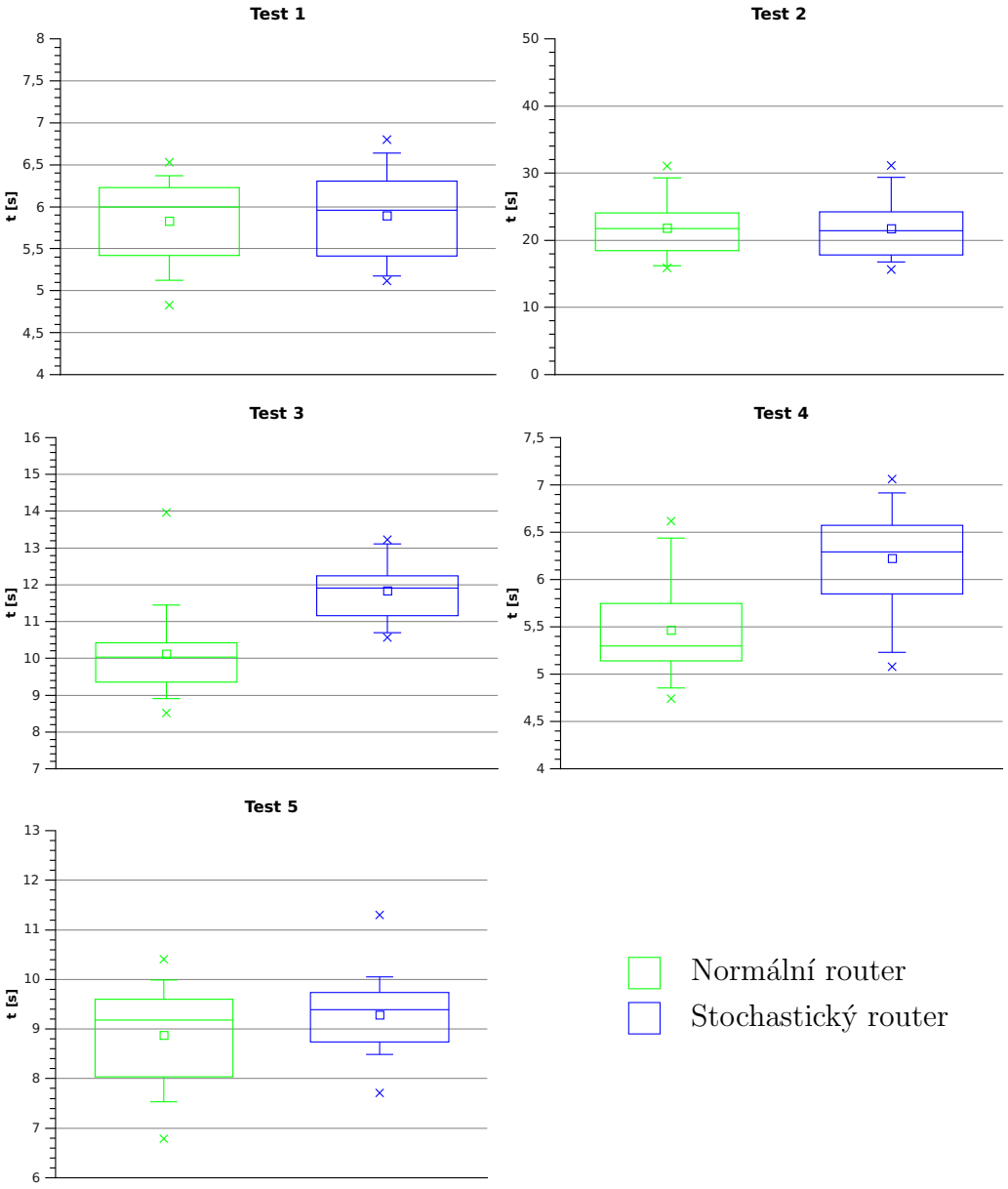


Obr. 8.5: Závislost počtu stojících vozidel na typu navigace

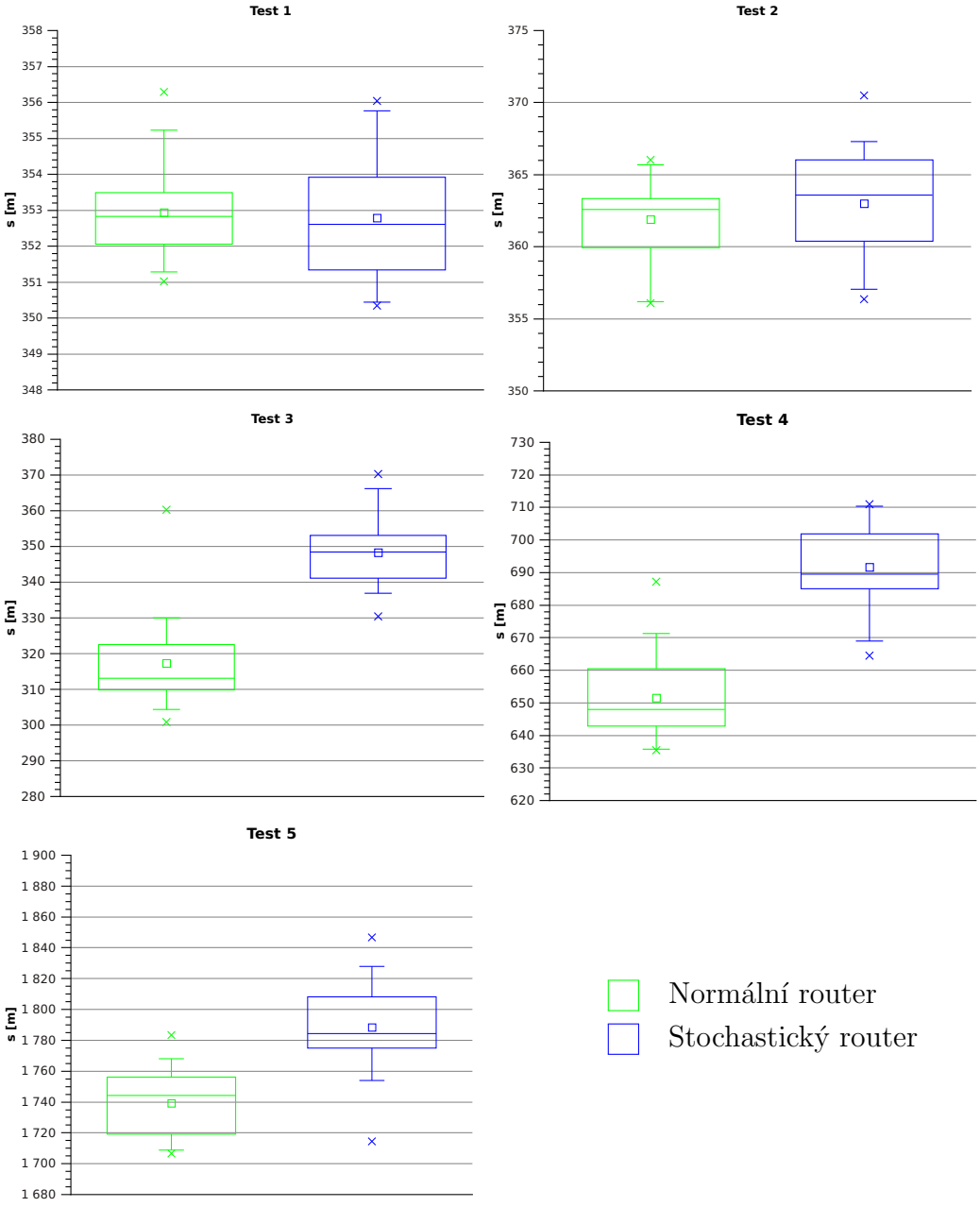
Následující parametry není možné vyhodnotit pro vozidla bez znalosti dopravní situace. Tato vozidla se zastaví v místě simulované dopravní komplikace a zůstanou zde stát po celou dobu simulace. Následně vytvořené vozidla díky neznalosti dopravní situace zvolí stejnou trasu a opět uváznou ve vzniklé zácpě. Tato kolona se vytvoří až po místo kde se vozidla tvoří a zablokují tvorbu dalších vozidel. Individuální statistiky se do výstupního souboru pro tyto vozidla nezapíší, jelikož k uložení dochází ve chvíli, kdy dorazí do cíle. Z toho důvodu jsou dále porovnávány pouze vozidla se znalostí dopravní situace.



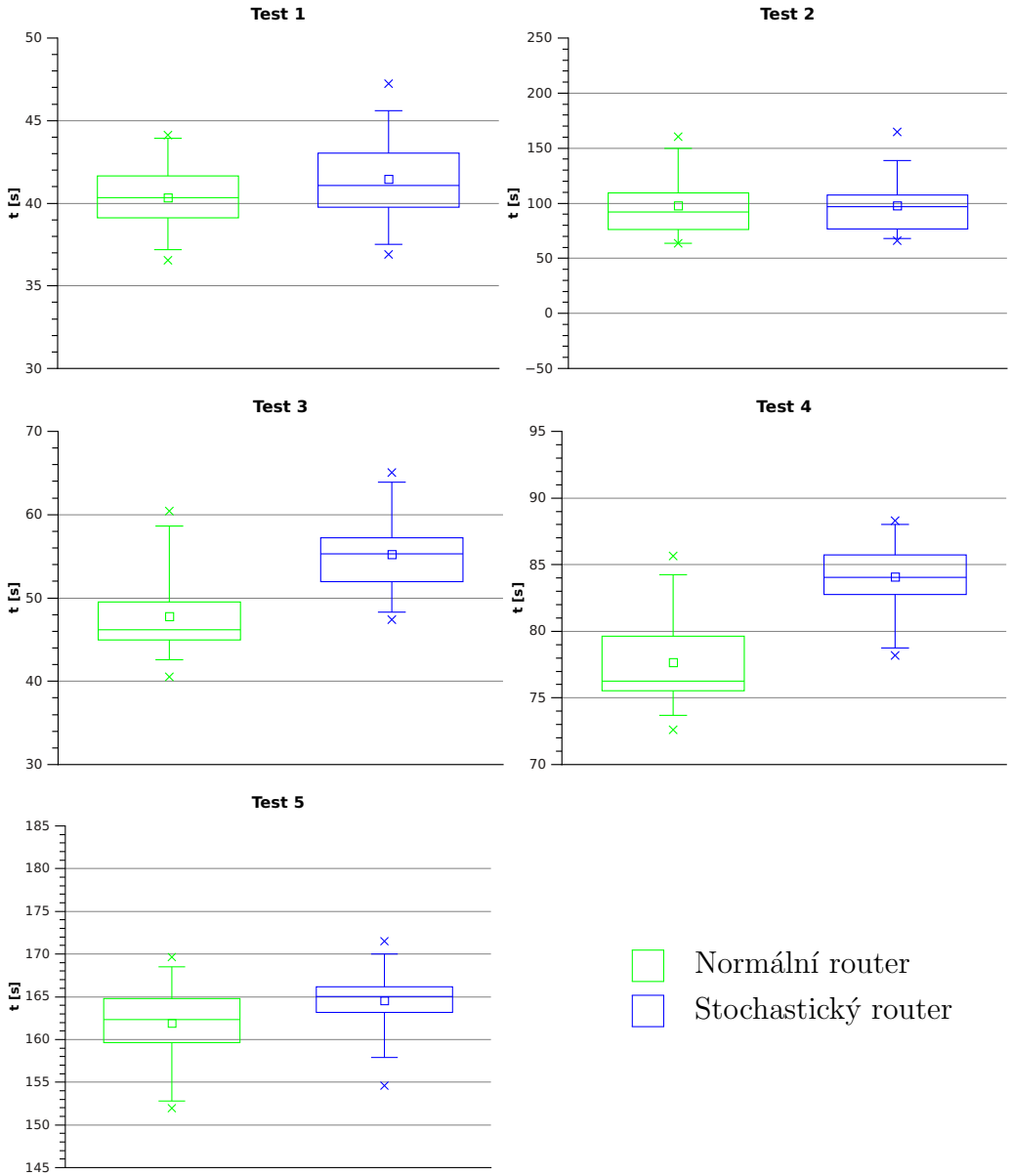
Obr. 8.6: Závislost doby stání vozidel na typu navigace



Obr. 8.7: Závislost doby pomalé jízdy vozidel na typu navigace



Obr. 8.8: Závislost ujeté vzdálenosti vozidel na typu navigace



Obr. 8.9: Závislost doby jízdy vozidel na typu navigace

Z grafů 8.4 a 8.5 je patrné, že vozidla se znalostí dopravní situace dosahují výrazně lepších výsledků. Doba průjezdu tisíce vozidel je téměř dvojnásobná než u vozidel bez znalosti situace, což je způsobeno především tvorbou kolon a zablokováním hlavní silnice na mapě. Počet stojících vozidel je ze stejného důvodu až několikanásobně vyšší než u vozidel se znalostí dopravní situace.

Stochastická navigace oproti běžnému hledání trasy vykazuje ve většině měřených parametrů a na většině map mírně horší výsledky.

8.3.1 Ověření nulové hypotézy

Pro ověření nulové hypotézy, že stochastická navigace vykazuje srovnatelné výsledky jako navigace normální, byl proveden nepárový Mann-Whitneyho test.

Z Mann–Whitneyho testu na obrázku 8.10 vyplývá, že pro mapy *Test 3* a *Test 4* je možné na hladině významnosti $\alpha = 0,01$ (zelená čára v grafech) jednostranně zamítnout nulovou hypotézu, že jsou výsledky pro normální i stochastickou navigaci totožné a přijmout alternativní, že normální navigace je lepší. Kromě parametru *čas průjezdu 1000 vozidel*, který je možné zamítnout na mapě *Test 3* s hladinou významnosti $\alpha = 0,05$ (červená čára v grafech). Pro mapu *Test 4* není možné zamítnout nulovou hypotézu pro parametr *čas průjezdu 1000 vozidel*. Na základě grafů 8.5 až 8.9 je tedy možné přijmout alternativní hypotézu, že kromě jednoho parametru dosahuje stochastická navigace horších výsledků než navigace normální.

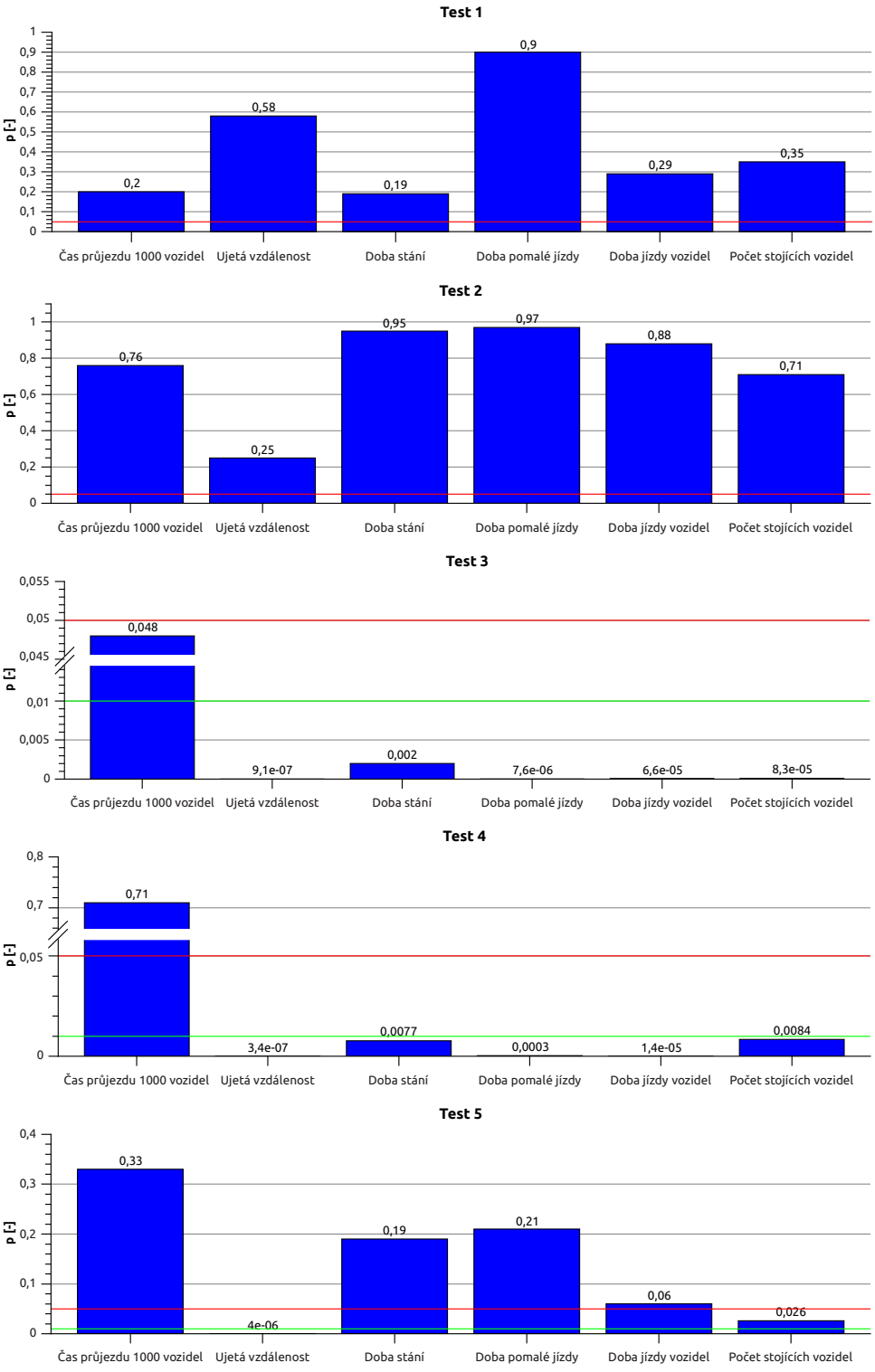
Pro mapu *Test 5* je možné nulovou hypotézu jednostranně zamítnout pouze pro parametr *ujetá vzdálenost* na hladině významnosti $\alpha = 0,01$ a parametr *počet stojících vozidel* na hladině významnosti $\alpha = 0,05$. Dle grafu 8.5 lze přijmout alternativní hypotézu, že stochastická navigace dosahuje lepšího výsledku než navigace normální pro parametr *počet stojících vozidel*. Dle grafu 8.8 je možné přijmout alternativní hypotézu, že stochastická navigace dosahuje horšího výsledku než navigace normální pro parametr *ujetá vzdálenost*. V ostatních parametrech jsou výsledky obou typů navigace statisticky srovnatelné.

Naopak pro mapy *Test 1* a *Test 2* není možné nulovou hypotézu zamítnou pro žádný měřený parametr. Je tedy možné potvrdit nulovou hypotézu, že oba typy navigace na těchto mapách dosahují srovnatelných výsledků pro všechny měřené parametry.

Výsledky jsou shrnuty v tabulce 8.1.

Tab. 8.1: Shrnutí výsledků Mann-Whitneyho testu porovnávajícího normální (N) a stochastickou (S) navigaci. Údaj v závorce udává hladinu významnosti, na které byla přijata hypotéza, že je daná metoda navigace lepší.

	T1	T2	T3	T4	T5
Čas průjezdu 1000 vozidel	–	–	N (0,05)	–	–
Ujetá vzdálenost	–	–	N (0,01)	N (0,01)	N (0,01)
Doba stání	–	–	N (0,01)	N (0,01)	–
Doba pomalé jízdy	–	–	N (0,01)	N (0,01)	–
Doba jízdy vozidel	–	–	N (0,01)	N (0,01)	–
Počet stojících vozidel	–	–	N (0,01)	N (0,01)	S (0,05)

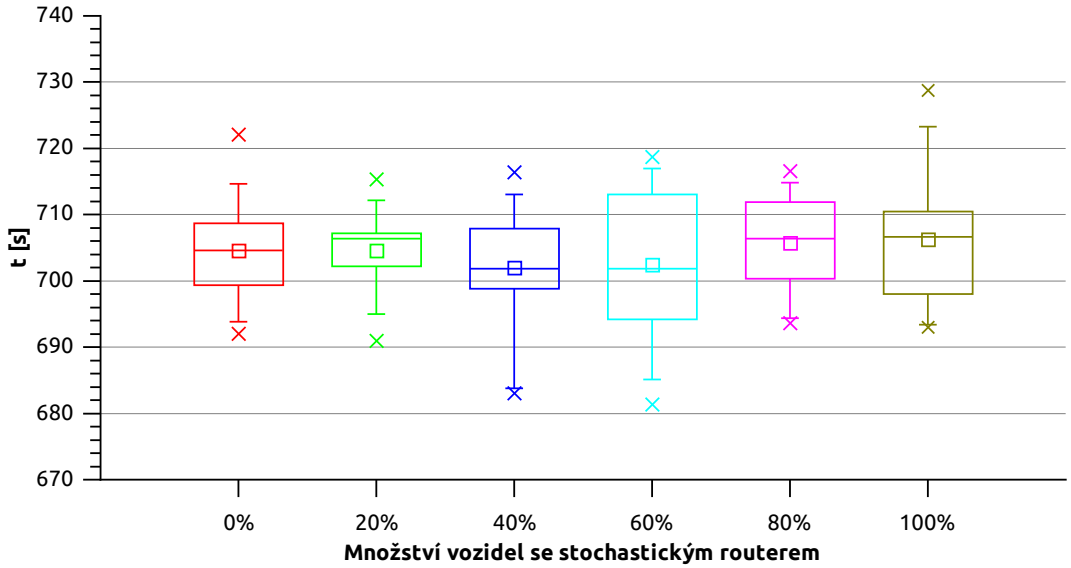


Obr. 8.10: Test Mann-Whitney pro normální a stochastický router pro jednotlivé mapy se zvýrazněnou hladinou významnosti $\alpha = 0,05$ (červená čára) a $\alpha = 0,01$ (zelená čára)

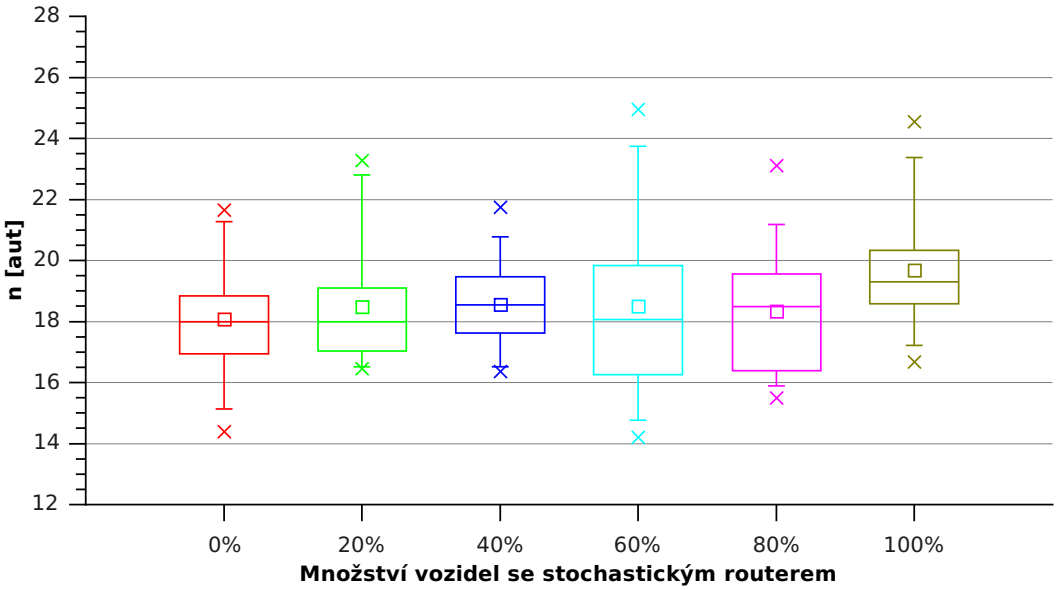
8.4 Různé koncentrace vozidel s určitým typem navigace

Tyto testy byly prováděny na mapě *Test 4* pro odlišné množství vozidel s určitým typem navigace.

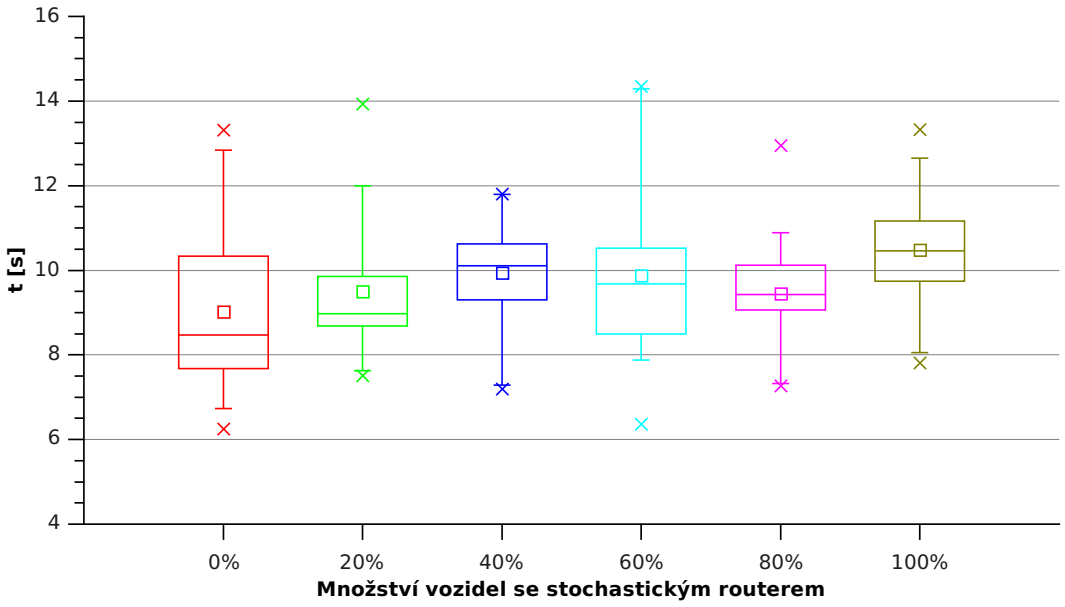
8.4.1 Různý poměr mezi normální a stochastickou navigací



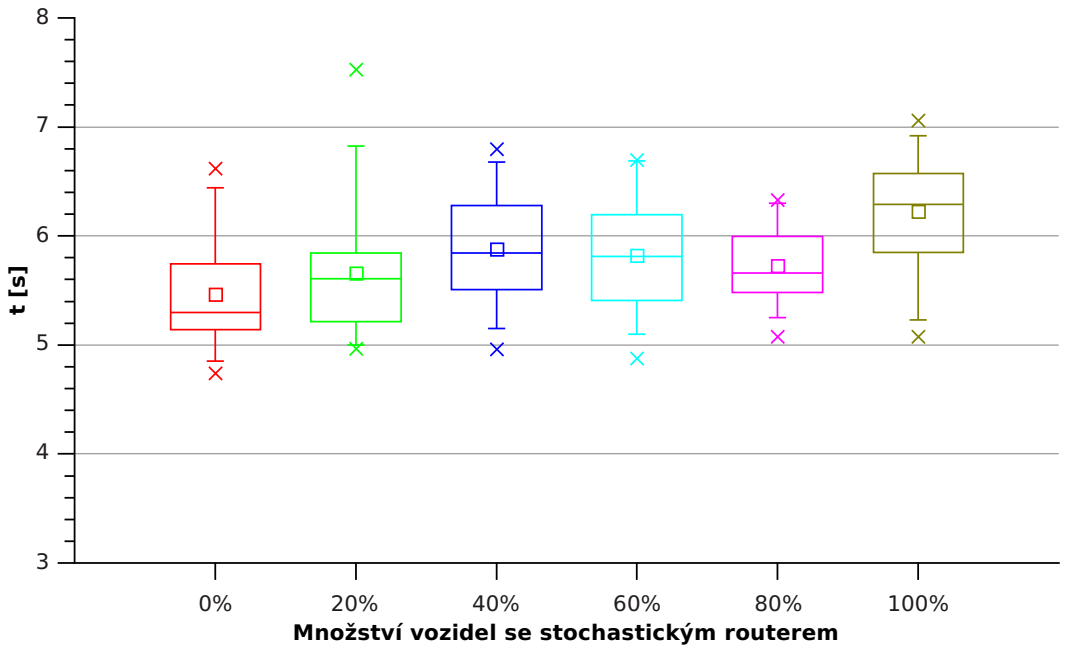
Obr. 8.11: Závislost doby průjezdu 1000 vozidel na typu navigace



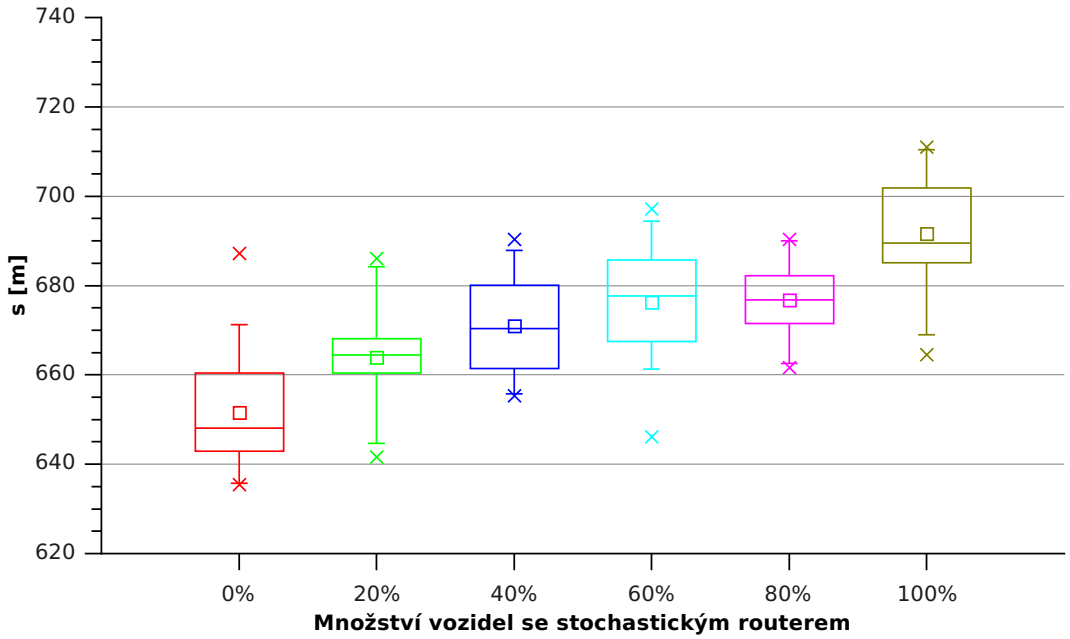
Obr. 8.12: Závislost počtu stojících vozidel na typu navigace



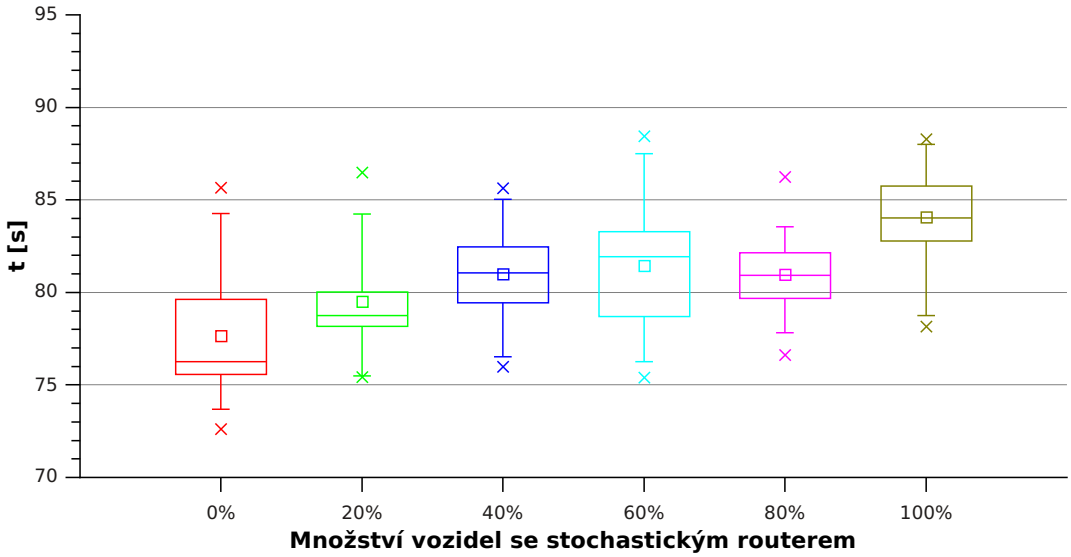
Obr. 8.13: Závislost doby stání vozidel na typu navigace



Obr. 8.14: Závislost doby pomalé jízdy vozidel na typu navigace



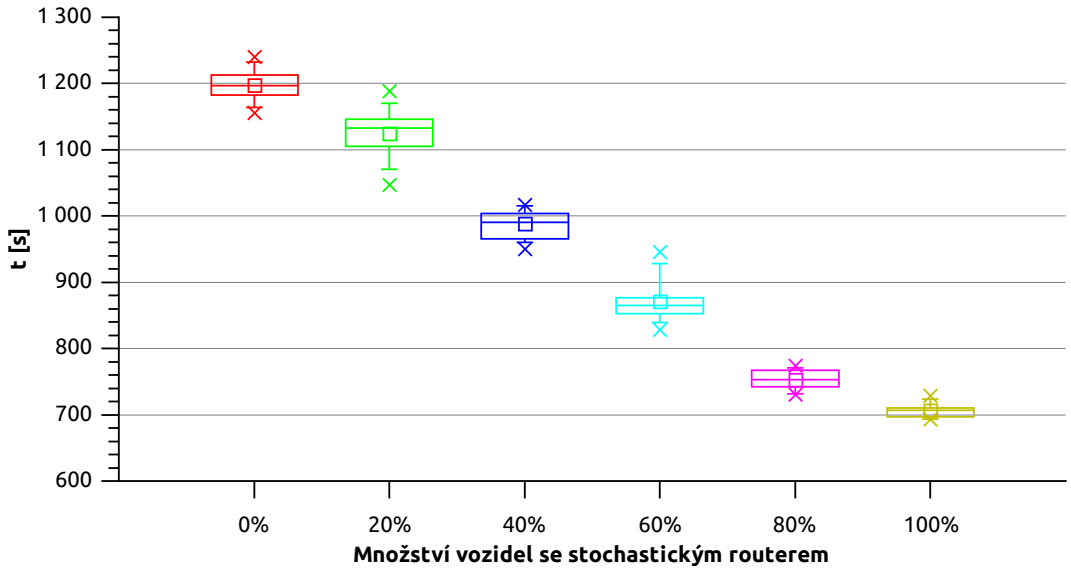
Obr. 8.15: Závislost ujeté vzdálenosti vozidel na typu navigace



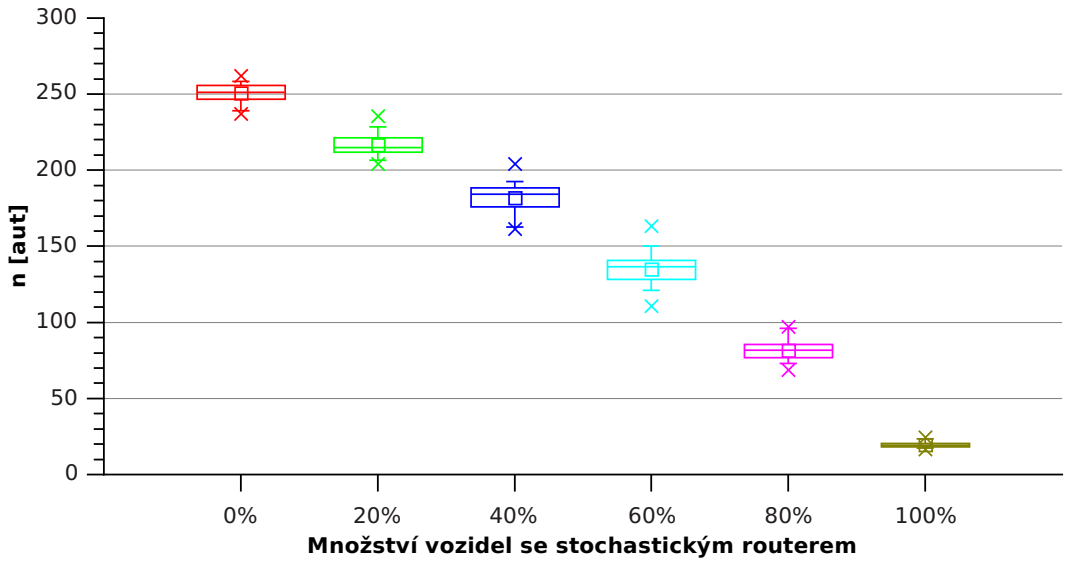
Obr. 8.16: Závislost doby jízdy vozidel na typu navigace

Z grafů 8.11 až 8.16 je patrné, že při změně množství aut se stochastickou navigací dochází k plynulé změně parametrů od nulové koncentrace vozidel se stochastickou navigací až po situaci, kdy jsou v mapě pouze vozidla se stochastickou navigací.

8.4.2 Různý poměr mezi stochastickou navigací a bez znalosti dopravní situace



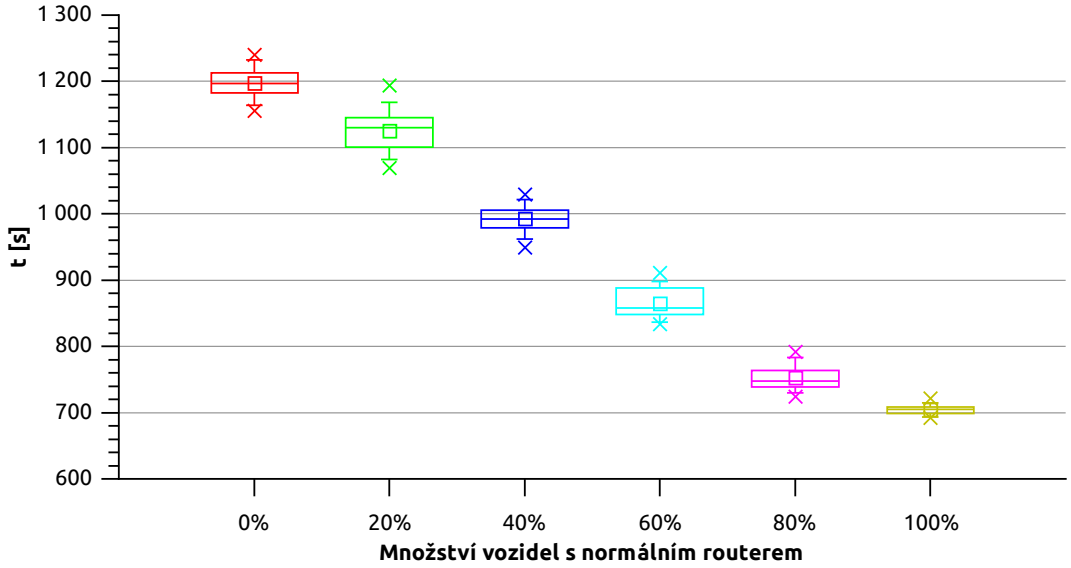
Obr. 8.17: Závislost doby průjezdu 1000 vozidel na množství vozidel se stochastickou navigací



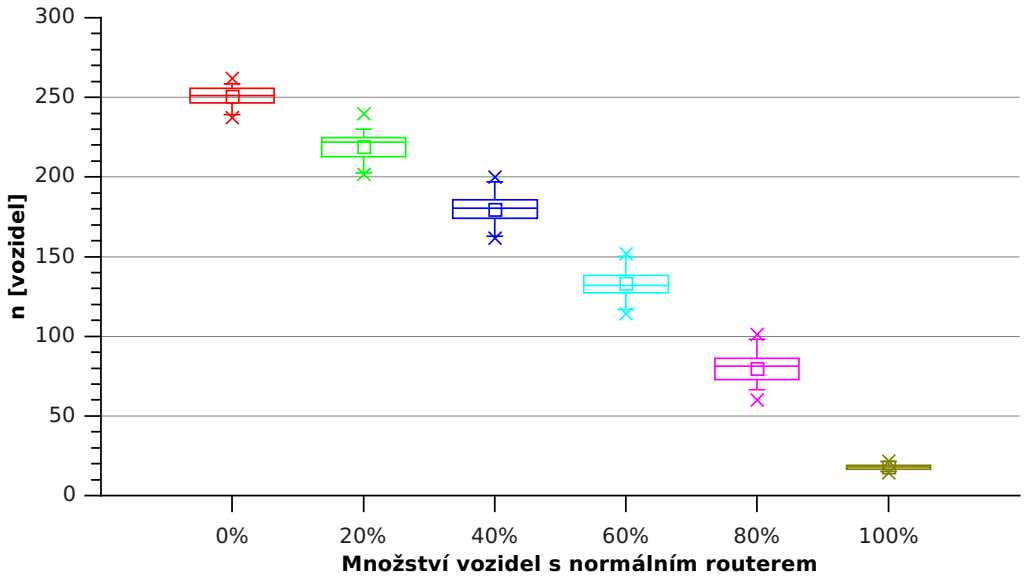
Obr. 8.18: Závislost počtu stojících vozidel na množství vozidel se stochastickou navigací

Ostatní parametry není možné vyhodnotit, jelikož nejsou dostupná data z vozidel bez znalosti dopravní situace, které uvázly v dopravní zácpě. Z grafů 8.17 a 8.18 je zřejmé, že čas průjezdu tisíce vozidel výrazně klesá se zvyšujícím se množstvím vozidel se stochastickou navigací. Čas je téměř dvojnásobně nižší, než když jsou v simulaci pouze vozidla bez znalosti dopravní situace. Také množství stojících vozidel výrazně klesá se zvyšujícím se množstvím vozidel se stochastickou navigací a to až 12×.

8.4.3 Různý poměr mezi normální navigací a bez znalosti dopravní situace



Obr. 8.19: Závislost doby průjezdu 1000 vozidel na množství vozidel s normální navigací



Obr. 8.20: Závislost počtu stojících vozidel na množství vozidel s normální navigací

Opět není možné vyhodnotit ostatní parametry, jelikož nejsou dostupná data z vozidel bez znalosti dopravní situace, které uvázly v dopravní zácpě. Podobně

jako v předchozí kapitole je z grafů 8.19 a 8.20 patrné, že čas průjezdu tisíce vozidel i množství stojících vozidel výrazně klesá se zvyšujícím se množstvím vozidel s normální navigací.

9 ZÁVĚR

Během řešení diplomové práce byly studovány algoritmy pro hledání cesty v orientovaném grafu (topologické mapě). Do dopravního simulátoru TRASI byly implementovány algoritmy A* a Floyd-Warshall. Pomocí těchto algoritmů byla vytvořena navigace, která může každému autu sestavit trasu jízdy z výchozího bodu do cíle. Dále byly implementovány události k hlášení dopravní situace. Auta mohou hlásit *stání na místě*, *nízkou rychlost* nebo *dopravní hlášení*. Tyto události se předávají ostatním autům pomocí komunikační vrstvy. Byly implementovány vrstvy C2I a C2C. Komunikační vrstva C2I simuluje předávání zpráv pomocí infrastruktury tvořené sítí přijímačů a vysílačů. Komunikační vrstva C2C simuluje přímé bezdrátové předávání zpráv mezi vozidly. Byla také navržena a implementována nová navigační metoda – tzv. stochastická navigace. Tato navigace na základě dopravních událostí navrhne několik alternativních tras jízdy. Z nich, na základě informací o propustnosti dílčích cest daných tras, náhodně vybere jednu. Důsledkem je, že dvě auta na prakticky stejném místě ve stejném okamžiku se stejným cílovým bodem mohou od stejné navigace dostat odlišnou náhradní trasu. To by mělo rozložit dopravní zátěž v okolí nahlášené dopravní události a zabránit tak přetížení alternativní trasy.

Z výsledků testů vyplývá, že stochastická navigace oproti očekáváním nedosahuje lepších výsledků než navigace normální pro měřené parametry *Množství stojících vozidel*, *Doba průjezdu 1000 vozidel*, *Doba stání*, *Doba pomalé jízdy*, *Ujetá vzdálenost* a *Doba jízdy vozidel*. Pro mapy *Test 1* a *Test 2* jsou výsledky obou typů navigace srovnatelné. Na mapách *Test 3* a *Test 4* dosahuje normální navigace pro většinu parametrů lepších výsledků a to na hladině významnosti $\alpha = 0,01$. Stochastická navigace vykazuje lepší výsledek pouze na mapě *Test 5* pro měřený parametr *Počet stojících vozidel*. Nebylo tedy prokázáno, že by stochastická navigace zabraňovala přetížení cest v okolí dopravní události a přispívala tak ke zlepšení dopravní situace a celkově lepší propustnosti dané oblasti.

LITERATURA

[1] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; aj.: *Introduction to algorithms*. Cambridge, Massachusetts: The MIT Press, třetí vydání, 2009, ISBN 9780262033848, 1313 s.

[2] Dym, C: *Principles of mathematical modeling*, ročník 13. Harvey Mudd College, Claremont, California, U.S.A.: Elsevier Science, druhé vydání, 2004, ISBN 9780080470283, 151–173 s.

[3] Honzík, Petr: *GA102/09/1897 - Bezpečnost automobilové dopravy - BAD*. Grantová agentura České republiky, [online].
Dostupné z WWW: <http://www.isvav.cz/projectDetail.do?rowId=GA102/09/1897>.

[4] Honzík, Petr: *Strojové učení*. Brno: FEKT Vysokého učení technického v Brně, 2006, 85 s.

[5] Hynčica, Tomáš: *A New Traffic and Vehicular Communication Simulator TRASI. Student EEICT*, 2013: s. 72–76.

[6] Hynčica, Tomáš; Hynčica, Ondřej; Kučera, Lukáš: *Simulation analysis of urban traffic flows with alternate routes. 17th International conference on soft computing MENDEL*, 2011: s. 282–288.

[7] Kučera, Lukáš: *Vývoj dopravního simulátoru – objekt křižovatka*. Bakalářská práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2010, vedoucí práce byl Ing. Petr Honzík Ph.D.

[8] Muzika, Dávid: *Inteligentní import mapových podkladů do TRASI*. Diplomová práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2013, vedoucí práce byl Ing. Petr Honzík Ph.D.

[9] Patel, Amit: *Amit's A* Pages*. 2012, [online], [cit. 2012-03-06].
Dostupné z WWW: <http://theory.stanford.edu/~amitp/GameProgramming/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

TRASI	Traffic simulator
FW	Floyd-Warshall
GA ČR	Grantová agentura České republiky
GUI	grafické uživatelské rozhraní
C2C	Car to Car
C2I	Car to Infrastructure
FIFO	First-in, First-out